The 11th Conference of PhD Students in Computer Science

Volume of short papers

CS^2

Organized by the Institute of Informatics of the University of Szeged

June 25 – June 27, 2018
Szeged, Hungary

Scientific Committee:

János Csirik (Co-Chair, SZTE)
Lajos Rónyai (Co-Chair, SZTAKI, BME)
András Benczúr (ELTE)
András Benczúr (SZTAKI)
Hassan Charaf (BME)
Tibor Csendes (SZTE)
László Cser (BCE)
Erzsébet Csuhaj-Varjú (ELTE)
József Dombi (SZTE)
István Fazekas (DE)
Zoltán Fülöp (SZTE)
Aurél Galántai (ÓE)
Zoltán Gingl (SZTE)
Tibor Gyimóthy (SZTE)
Katalin Hangos (PE)
Zoltán Horváth (ELTE)
Márk Jelasity (SZTE)
Zoltán Kása (Sapientia EMTE)
László Kóczy (SZE)
János Levendovszki (BME)
Gyöngyvér Márton (Sapientia EMTE)
Branko Milosavljevic (UNS)
Valerie Novitzka (TUKE)
László Nyúl (SZTE)
Marius Otesteanu (UPT)
Attila Pethő (DE)
Vlado Stankovski (UNILJ)
Tamás Szirányi (SZTAKI)
Péter Szolgay (PPKE)
János Sztrik (DE)
János Tapolcai (BME)
János Végh (ME)
Daniela Zaharie (UVT)

Organizing Committee:
Attila Kertész, Balázs Bánhelyi, Tamás Gergely, Zoltán Kincses
Address of the Organizing Committee
c/o. Attila Kertész
University of Szeged, Institute of Informatics
H-6701 Szeged, P.O. Box 652, Hungary
Phone: +36 62 546 396, Fax: +36 62 546 397
E-mail: cscs@inf.u-szeged.hu
URL: http://www.inf.u-szeged.hu/~cscs/

Sponsors
Supported by the project "Integrated program for training new generation of scientists in the fields of computer science", No. EFOP-3.6.3-VEKOP-16-2017-00002. The project has been supported by the European Union and co-funded by the European Social Fund.
University of Szeged, Institute of Informatics
Polygon Publisher
Association of Hungarian PhD and DLA Students, Scientific Section of Mathematics and Informatics

Preface

This conference is the eleventh in a series. The organizers aimed to bring together PhD students working on any field of computer science and its applications to help them publishing one of their first abstracts and papers, and provide an opportunity to hold a scientific talk. As far as we know, this is one of the few such conferences. The aims of the scientific meeting were determined on the council meeting of the Hungarian PhD Schools in Informatics: it should

- provide a forum for PhD students in computer science to discuss their ideas and research results;
- give a possibility to have constructive criticism before they present the results at professional conferences;
- promote the publication of their results in the form of fully refereed journal articles; and finally,
- promote hopefully fruitful research collaboration among the participants.

The papers emerging from the presented talks will be invited to be considered for full paper publication the Acta Cybernetica journal.

The organizers hope that the conference will be a valuable contribution to the research of the participants, and wish a pleasant stay in Szeged.

Szeged, June 2018

Attila Kertész
Balázs Bánhelyi
Tamás Gergely
Zoltán Kincses

Contents

Program

Monday, June 25

08:00 – 09:00 Registration

09:00 – 09:15 Opening

09:15 – 10:35 Talks – Artifical Intelligence (4x20 min.), Static Analysis (4x20 min.)

10:35 – 11:00 Break

11:00 – 12:40 Talks – Cloud Computing I. (5x20 min.), Testing (4x20 min.)

12:40 – 14:00 Lunch

14:00 – 14:50 Plenary Talk

14:50 – 15:10 Break

15:10 – 16:30 Talks – Cloud Computing II. (4x20 min.), Image Processing I. (4x20 min.)

16:30 – 18:10 Talks – Education (3x20 min.), Image Processing II. (4x20 min.)

19:00 – 21:00 Reception at the Rector's Building

Tuesday, June 26

09:00 – 09:15 Registration

08:30 – 10:35 Talks – Optimization (4x20 min.)

10:35 – 11:00 Break

11:00 – 13:00 Talks – Algorithms (6x20 min.)

13:00 – 14:00 Lunch

14:00 – 14:50 Plenary Talk

15:00 – 19:00 Social program

19:00 – 22:00 Gala Dinner at the Rector's Building

Wednesday, June 27

09:00 – 09:15	Registration
09:15 – 10:35	Talks – Programming Languages (4x20 min.)
10:35 – 11:00	Break
11:00 – 11:50	Plenary Talk
11:50 – 12:00	Break
12:00 – 13:00	Talks – Evaluation (3x20 min.)
13:00 – 14:00	Lunch
14:00 – 15:00	Talks – Business Process (3x20 min.)
15:00 – 15:30	Closing

Detailed program

Monday, June 25

08:30	**Registration**
09:00	**Opening**
Session 1	**Artificial Intelligence** - Session chair: Márk Jelasity, Room: Szőkefalvi-Nagy
09:15	Gergely Pap, Tamás Grósz, László Tóth: *Semi-Supervised Training of Cell-Classifier Neural Networks*
09:35	László Tóth: *Preliminary Concepts for Requirements Mining and Classification using Hidden Markov Model*
09:55	Yangyuan Li, Tien Van Do: *Long Short-term Memory Recurrent Neural Networks Models to Forecast the Resource Usage of MapReduce Applications*
10:15	Ádám Budai, Kristóf Csorba: *Deep Reinforcement Learning: A Study of the CartPole Problem*
Session 2	**Static Analysis** - session chair: Judit Jász, Room: Vályi
09:15	Péter Gál, Edit Pengő: *Primitive Enthusiasm: A Road to Primitive Obsession*
09:35	András Kicsi, Viktor Csuvik: *Feature Level Metrics Based on Size and Similarity in Software Product Line Adoption*
09:55	Edit Pengő, Zoltán Ságodi, Ervin Kóbor: *Who Are You not gonna Call? A Definitive Comparison of Java Static Call Graph Creator Tools*
10:15	Tibor Brunner, Péter Szécsi, Zoltan Porkoláb: *Bug Path Reduction Strategies for Symbolic Execution*
10:35	Break
Session 3	**Cloud Computing I.** - session chair: Richárd Farkas, Room: Szőkefalvi-Nagy
11:00	Biswajeeban Mishra: *Evaluating Performance of MQTT Brokers*
11:20	Yangyuan Li, Tien Van Do: *Regression Models to Predict the Resource Usage of MapReduce Applications*
11:40	András Márkus, Attila Kertész: *Multi-Cloud Management Strategies for Simulating IoT Applications*
12:00	Binh Lam Van Thanh: *Should we Ignore the Practical Aspects in Modelling Server Clusters?*
12:20	Ádám Belákovics, Arnold Czémán, Imre Szeberényi: *Designing and Testing VM Allocation Algorithms for the CIRCLE Cloud Manager*
Session 4	**Testing** - session chair: Ákos Kiss, Room: Vályi
11:00	Dániel Lukács, Gergely Pongrácz, Máté Tejfel: *Keeping P4 Switches Fast and Fault-free through Automatic Verification*
11:20	Gabriella Tóth, Máté Tejfel: *Axiom-based Property Verification for P4 Programs*
11:40	Nadera Aljawabrah, Tamás Gergely: *Visualization of Test-to-code Relations to Detect Problems of Unit Tests*
12:00	Gábor Horváth, Réka Kovács, Péter Szécsi: *Towards Proper Differential Analysis of Static Analysis Engine Changes*
12:40	Lunch
14:00	**Plenary Talk** - Room: Bolyai Bálint Daróczy: *Riemann manifolds and hierarchical structures*
14:50	Break

Session 5	**Cloud Computing II.** - session chair: Tamás Vinkó, Room: Szőkefalvi-Nagy
15:10	Andrea Huszti, Norbert Oláh: *Identity-Based Cloud Authentication Protocol*
15:30	Zoltán Richárd Jánki, Vilmos Bilicki: *Full-stack FHIR-based MBaaS with Server- and Client-side Caching Capable WebDAO*
15:50	Zoltán Szabó, Vilmos Bilicki: *FHIR-based Healthcare System Backend with Deep Cloud Side Security*
16:10	Attila Selmeci, István Orosz: *Software as a Service Operation Model in Cloud Based ERP Systems*
Session 6	**Image Processing I.** - session chair: Gábor Németh, Room: Vályi
15:10	Judit Szűcs, Péter Balázs: *Strip Constrained Binary Tomography*
15:30	Gábor Lékó, Péter Balázs, László Varga: *Projection Selection with Sequential Selection Methods using Different Evaluation Measures*
15:50	Ákos Tóth, Roland Kunkli: *An Approximative and Semi-automated Method to Create MPEG-4 Compliant Human Face Models*
16:10	Csaba Bálint, Gábor Valasek: *Operations on Signed Distance Functions*
16:30	Break
Session 7	**Education** - session chair: József Dániel Dombi, Room: Szőkefalvi-Nagy
16:50	Sándor Balázs Domonkos, Tamás Németh: *Use data mining methods in quality measurement in the education systems*
17:10	Chaman Verma, Veronika Stoffová, Zoltán Illés, Sanjay Dahiya: *Gender classification Model of European school student's towards Computer Learning*
17:30	Enikő Ilyés: *Agile method in education*
Session 8	**Image Processing II.** - session chair: Péter Balázs, Room: Vályi
16:50	György Kalmár, Alexandra Büki, Gabriella Kékesi, Gyöngyi Horváth, László G. Nyúl: *Feature Extraction and Classification for Pupillary Images of Rats*
17:10	Márton Véges and Viktor Varga: *Monocular Estimation of 3D Poses from a Distance*
17:30	Viktor Varga and Márton Véges: *Exploiting Temporal Context in 2d to 3d Human Pose Regression*
17:50	Máté Csákvári and András Sárkány: *Towards the Understanding of Object Manipulations by Means of Combining Common Sense Rules and Deep Networks*
18:10	Free program
19:00	**Reception** at the Rector's Building

Tuesday, June 28

09:00	Registration
Session 9	**Optimization** - session chair: Boglárka Gazdag-Tóth, Room: Szőkefalvi-Nagy
09:15	Viktor Homolya: *Graph-based Analysis of Influence Spread*
09:35	Péter Hudoba, Péter Burcsi: *Multi Party Computation Motivated by the Birthday Problem*
09:55	Abigél Mester, Emilia Heinz, Balázs Bánhelyi, Elvira D. Antal, Edit Mikóné Jónás, József Horváth, Tibor Csendes: *Decision Support Heuristic for Dairy Farms*
10:15	Zsolt Mihály, Zsombor Sentes, Zoltán Lelkes: *Ant Colony Optimization Based Algorithm For Solving Scheduling Problems with Setup Times on Parallel Machines*
10:35	Break
Session 10	**Algorithm** - session chair: Gábor Gosztolya, Room: Szőkefalvi-Nagy
11:00	Ilku Krisztián János, Judit Tamás: *Topology-based Classification Error Calculation based on IndoorGML Document*
11:20	Kitti Gelle, Szabolcs Iván: *Lookahead can Help in Maximal Matching*
11:40	Dénes Bartha: *Reconstruction of Rooted Directed Trees*
12:00	Dávid Nagy, Tamás Mihálydeák, László Aszalos: *Different Types of Search Algorithms for Rough Sets*
12:20	Szabolcs Szekér, Ágnes Vathy-Fogarassy: *Measuring the Similarity of two Cohorts in the n-dimensional Space*
12:40	Abrar Hussain, József Dombi: *Enhanced Adaptivity Fuzzy Control Using a New Type of Membership Function*
13:00	Lunch
14:00	**Plenary Talk** - Room: building of Hungarian Academy of Sciences, Somogyi utca 7. Michael Mackey: *Understanding, treating and avoiding hematological disease: Better medicine through mathematics?*
14:50	Free program
15:00	**Social Program** Visit the ELI-ALPS facility
19:00	**Gala Dinner** at the Rector's Building

Wednesday, June 29

09:00	Registration
Session 11	**Programming Languages** - session chair: Árpád Beszédes, Room: Szőkefalvi-Nagy
09:15	Sándor Bácsi, Gergely Mezei: *Towards a Classification to Facilitate the Design of Domain-Specific Visual Languages*
09:35	Zsolt Parragi, Zoltán Porkoláb: *Instantiation Context Aware Types in C++*
09:55	Norbert Luksa, Tamás Kozsik: *Parallelisation of Haskell Programs by Refactoring*
10:15	Péter Gál: *JavaScript-only Parallel Programming of Embedded Systems*
10:35	Break
11:00	**Plenary Talk** - Room: Bolyai Massimiliano Di Penta: *Empirical assessment of software engineering research: Pitfalls and solutions*
11:50	Break
Session 12	**Evaluation** - session chair: Zoltán Gingl, Room: Szőkefalvi-Nagy
12:00	Tibor Kovács, Gábor Simon, Gergely Mezei: *Benchmarking Graph Database Backends: What Works Well with Wikidata?*
12:20	Róbert Adrian Rill, Kinga Bettina Faragó: *Gaze-based Cursor Control Impairs Performance in Divided Attention*
12:40	Zsombor Paróczi: *LZ Based Compression Benchmark on PE Files*
13:00	**Lunch**
Session 13	**Business Process** - session chair: András London, Room: Szőkefalvi-Nagy
14:00	Dóra Mattyasovszky-Philipp, Bálint Molnár: *Cognitive Enterprise and Cognitive Information Systems*
14:20	Khawla Bouafia, Bálint Molnár: *Dynamic Business Process: Comparative Models and Workflow Patterns*
14:40	László Pusztai, Balázs Kocsi, István Budai, Lajos Nagy: *Industrial Process Modelling with Operations Research Method*
15:00	**Closing**
15:30	Farewall drinks

Riemann Manifolds and Hierarchical Structures

Bálint Daróczy
Institute for Computer Science and Control,
Hungarian Academy of Sciences (MTA SZTAKI), Hungary

We suggested in [1] the family of similarity kernels as a class of kernels applicable for classification and regression. The methods are capable of defining a single unified kernel even in the case of rich data types. The final kernels are define a generative model (Markov Random Field) on pairwise similarities and the Fisher information. The kernels do not depend on the parameters of the random graph and therefore we do not need to determine the relative importance of the basic modalities [2]. We extended the model for time-series classification in [1] and measuring similarity between subgraphs [3]. During the talk we consider the case of low order polynomials as Hamiltonians following the results in [4]. The expressive power of deep structures with a special family of metrics [5], extended Hessian metrics with proper quasi-arithmetic means, suggest us sparse representations and we investigate how the quality of approximation connected to the geometrical properties of the learning methods in certain hierarchical, deep structures without efficient "flattening".

References

[1] B. Daróczy, P. Vaderna, and A. Benczúr. *Machine learning based session drop prediction in LTE networks and its SON aspects.* In Proceedings of the 81st IEEE Vehicular Technology Conference (VTC Spring), 2015, Glasgow, Scotland, UK, pp. 1–5, IEEE

[2] B. Daróczy, D. Siklósi, R. Pálovics, and A. Benczúr. *Text Classification Kernels for Quality Prediction over the C3 Data Set.* In Proceedings of the 24th International Conference on World Wide Web (WWW'15), Florence, Italy, pp. 1441-1446, ACM

[3] J. Wachs, B. Daróczy, A. Hannák, K. Páll, and C. Riedl. *And Now for Something Completely Different: Visual Novelty in an Online Network of Designers.* accepted as full paper at the 10th ACM Conference on Web Science 2018, Amsterdam, Netherlands, ACM

[4] Henry W. Lin, Max Tegmark, and David Rolnick. *Why does deep and cheap learning work so well?* Journal of Statistical Physics 168(6), 2017, pp. 1223–1247, Springer

[5] Shun-ichi Amari and John Armstrong. *Curvature of Hessian manifolds.* Differential Geometry and its Applications 33, 2014, pp. 1–12, Elsevier

Understanding, Treating and Avoiding Hematological Disease: Better Medicine Through Mathematics?

Michael C. Mackey
Department of Physiology, McGill University, Canada

This talk will trace the use of mathematical models (typically framed as systems of differential delay equations) in conjunction with clinical and laboratory data in the development of our understanding of the origins of the dynamical disease cyclical neutropenia. I will then go on to outline how the modeling and understanding led to the development of effective treatments of the disease. The last part of the talk with outline how we have used that insight to avoid other hematological problems associated with the all too common administration of chemotherapy.

Empirical Assessment of Software Engineering Research: Pitfalls and Solutions

Massimiliano Di Penta
Department of Engineering, University of Sannio, Italy

The availability of a wide variety of software repositories, ranging from Questions and Answer forums to mailing lists, forges and issue trackers opens the road for building recommender systems aimed at supporting developers in their activities. Upon evaluating such recommenders, in most cases researchers focus on the underlying approach capability of providing accurate and complete results. This is insufficient because a tool could achieve an almost perfect precision and recall, however, its provided features do not really help software developers in their everyday's tasks.

This keynote will report my personal experience (and those of my collaborators) in evaluating recommenders, showing that an offline evaluation of the approach precision and recall is only a very preliminary starting point. Importantly, different kinds of (qualitative and quantitative) evaluations, having different size and level of control, and above all involving humans, are required to achieve results able to convince practitioners of the actual usefulness and applicability of a tool. Moreover, I will discuss and emphasize the important role played by contextual information in the evaluation of recommender systems.

Decision support heuristic for dairy farms

Abigél Mester, Emília Heinz, Balázs Bánhelyi, Elvira D. Antal, Edit Mikóné Jónás, József Horváth, Tibor Csendes

Abstract: After having a smart phone based microsimulation tool for the optimal decision to be made on selling/keeping the ill cow (mastitis) last year, we have started a new applied research project to improve the quality of the decision and the profitability. We can get improvement by utilizing local data of the given dairy farm instead of national average values of the critical parameters such as chances to get the illness again, length of the dry and productive periods etc. We report on the preliminary profitability improvement results. This time we take into consideration the lactation curve, and we also utilize the amount of produced milk as a basis of decision.

Keywords: milk production, mastitis, profitability, stochastic optimization, microsimulation

Introduction

One of the most unpredictable, the most profit sensitive sector of the common section of agriculture and economy is milk production. Since the anticipated milk price is volatile, we have to design scalable models to get efficient solutions [5]. According to some studies it is gainful to make predictions [7]. In the present study we investigate the possibility of effective economic modelling of an important decision: when to sell the cow after a diagnosed new mastitis illness. After visiting a few local farms, and getting to know more about the problem, we can say, that usually they sell a cow when it is in a very bad shape.

So far it looks like a cow is treated with the proper medicine once it gets ill with mastitis, and it is kept - if it is not in a really bad shape that it has to be sold. Using some mathematics, simulations and programming we can estimate the expected profit of an animal if we keep it or sell it. This way farmers do not have to keep unprofitable cows.

The whole planned research will last for years. First step is to show that the realistic based conception has it's own limitations. Later we plan to extend the system to a data mining and decision support system based on sophisticated method. We shall also complete our model to incorporate the related connecting economic subsystems such as the animal food production and milk processing.

Material and method

Material

A project like this requires close cooperation between programmers and agriculture workers. While we were collecting data for our study, we have visited a few diary farms. Using the most important factors of a cow's life we have built a pretty simple but hopefully detailed enough model.

With just a few information we can simulate the future of a cow. The starting data includes some data about the cow from the age to the illness number. And we also calculated some probabilities to make our decisions more realistic from some historical data. For example with the higher number of mastitis it will be more likely that a cow gets ill again. Although the milk production of a cow follows a specific curve, the dairy cycle curve, according to our computational tests, to optimise the purchasing decision, we can assume that the milk production is constant within the dairy cycle.

Let's look at some simple, specific data about mastitis as an example. We assume that the actual mastitis requires 5 days of healing with a probability of 70%, and 10 days with probability 30%. An additional interval of 15 days is needed to first profit from the milk production. To get ill, we have a daily probability of just 0.05% if it will be the first mastitis of the given cow, 0.1% for the second, 0.2% for the third, and 0.4% probability for all the later illnesses.

All of the profits in different states of the cow can be calculated from the data which is collected in diary farms, or using dispersion, or even using the daily data from local diary farms' databases.

Method

With our microsimulation model we investigate the possible best way to decide when to sell the ill cow. The basis of our technique is to simulate the life of a cow on daily basis. In other words, we start with a cow of a given age, number of already suffered mastitis illness, and in a given phase of the dairy cycle. For each day we check a list of possible event in the life of a cow. If an event is possible then we generate a random number to simulate a realistic experiment. This way we generate a list of events for the rest of the cow's life. So we can manage multiple events at the same time. For example we can keep on counting the lactation cycle days, while the cow is ill. While computing the possible events we use the same which were used during the calculation of our data. So the methods to determine the possible events can rely on fixed data, probability dispersion, or even a full list of historical data.

A few regularities can be discovered, like after a proper dry period, the milk production will resume. The cycle of that cow ends by its selling. The date of the purchase can be determined by our simple rule of thumb: we sell the cow if it reaches either the 6th mastitis, or its 10th year of living.

Having a model for the financial description for a cow, we simulate 100 times the possible outcome to have an approximate stochastic description of the distribution function of the profit. Then we can determine an optimal decision on the expected achievable profit. This microsimulation approach is similar to that used to investigate whether a time based ticket system is better than the existing trip based on in public transportation in Szeged [1, 3]. The coding was made in Java language, and the simulation programs were run on a blade server.

Results

By using our simulations we can predict the future of a cow, as you can see it in Figure 1.

Object oriented Java implementation not only makes it easier to supplement our program with newer and newer methods, but we can easily connect the algorithm with an Android implementation.

The simple program that is capable to solve such problems with straightforward input data is available for smart phones and tablets (having Android 6.0 or newer operating systems) at www.inf.u-szeged.hu/∼banhelyi/Buu

We shall update it regularly, and we also plan to implement the application in such a way that also earlier versions of Android should run it.

Figure 2. presents a sample of our prototype:

Figure 1: Cumulated profit of a cow in HUF according to the days spent in the farm. The minimal, maximal, average and 2 further quartiles curves of the distribution are depicted. These results were obtained based on 100 independent simulations of the probabilistic events in the model.

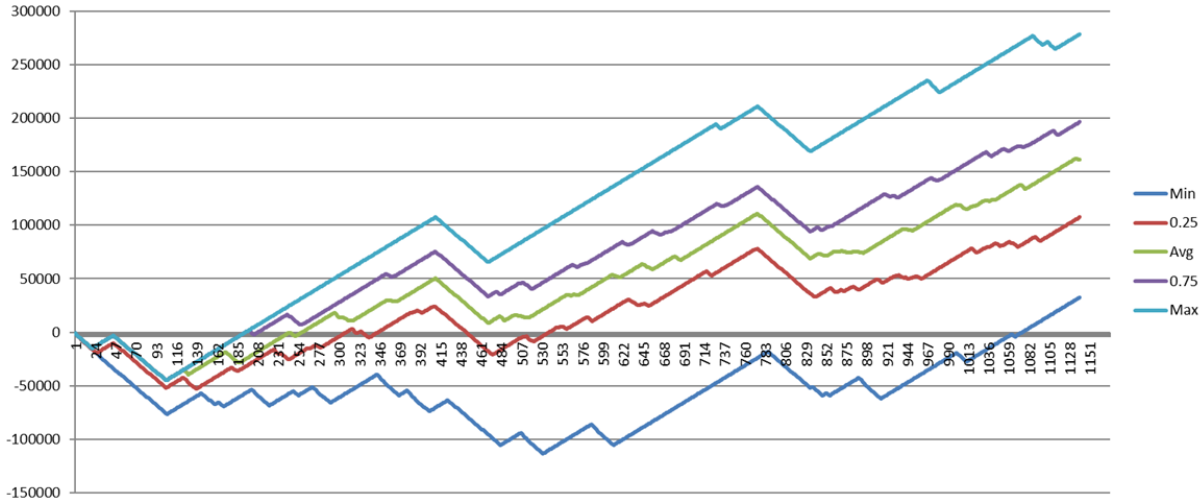
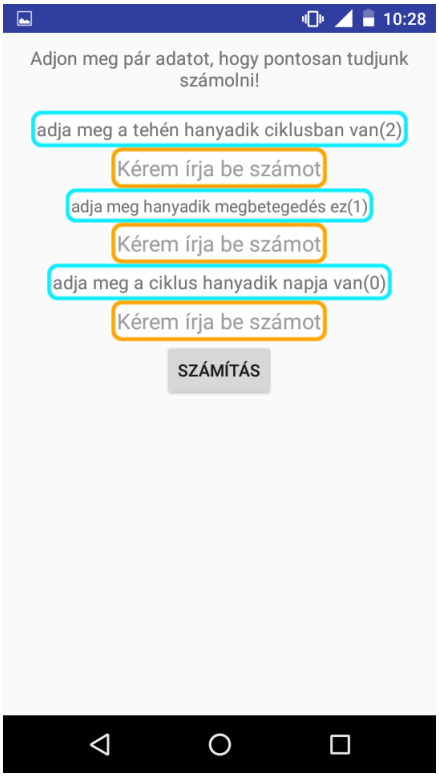


Figure 2: It is easy to add all of the input data using the Android program.



Acknowledgements

The authors are grateful for the android application coding and testing support given by Kristóf Mátéffy and Zsombor Gellérfi.

This work was supported by the project "Integrated program for training new generation of scientists in the fields of computer science", no. EFOP-3.6.3-VEKOP-16-2017-0002. The project

has been supported by the European Union and co-funded by the European Social Fund.

References

[1] Bernát Almási, Endre Palatinus. COMPUTATIONAL MODELLING OF THE ECONOMIC EFFECT OF THE TRAVEL TIME BASED TICKET SYSTEM, *Student Research Competition (TDK), University of Szeged*, 2010.

[2] Balázs Bánhelyi, Tibor Csendes, Abigél Mester, Edit Mikóné Jónás, and József Horváth. WHEN TO SELL THE COW?, *Review on Agriculture and Rural Development*, 2017.

[3] Balázs Blázsik, Tibor Csendes. ECONOMIC MODELLING OF THE TRANSITION FROM A TRAVEL TIMES BASED TO A TRAVEL TIME BASED TICKET SYSTEM, *Research Report for the Szeged Transportation Inc., KNRet*, 2010.

[4] Tibor Csendes, László Pál, Oscar H. Sendín, Julio R. Banga. THE GLOBAL OPTIMIZATION METHOD REVISITED, *Optimimization Letters Volume 2, pp. 445-455*, 2008.

[5] Pat Dillon, Thia Henessy, Laurence Shalloo, Fiona Thorne, Brendan Horan. FUTURE OUTLOOK FOR THE IRISH DIARY INDUSTRY: A STUDY OF INTERNATIONAL COMPETITIVENESS, INFLUENCE OF INTERNATIONAL TRADE REFORM AND REQUIREMENT FOR CHANGE, *International Journal of Diary Technology, Volume 61, Issue 1, pp. 16-29.*, 2008.

[6] Zsuzsanna Fekete, RoSwitha Baumung, Birgit Fuerst-Waltl, Krisztián Keller, Ferenc Szabó. THE EFFECT OF MILK YIELD ON THE PROFITABILITY AND ECONOMIC WEIGHT OF SELECTED TRAITS, *Zuchtungskunde, Volume 84, Issue 6, pp. 463-473.*, 2012.

[7] Margaret D. March, Laurence Shalloo, David J. Roberts, Willie Ryan. FINANCIAL EVALUATION OF HOLSTEIN FRIESIAN STRAINS WITHIN COMPOSITE AND HOUESED UK DIARY SYSTEMS, *Livestock Science, Volume 200, pp. 14-22.*, 2017.

A new Approach to Fuzzy Control using Distending Function

Abrar Hussain, József Dombi

Abstract: The paper presents a novel design for fuzzy logic control using the Distending Function (DF) as a membership function. The proposed design is close to human input and can be transformed into fuzzy model for stability analysis using conventional control theory. The design procedure is simplified by employing fuzzy arithmetic and properties of this new type of membership function. Its compatible with the existing operators system and can be used with already developed fuzzy models to achieve greater flexibility in design process.The effectiveness of the proposed methodology is demonstrated by designing tracking fuzzy controllers for vehicle lateral dynamics and water tank systems.

Keywords: Adaptive fuzzy control, Distending function, Membership function

Introduction

Fuzzy theory has been an extensive area of research since its first development, nearly half century ago by Lotfi A. Zadeh [1] and is providing applications in various fields of life [2], [3]. In the field of control system design, complex ill defined non linear processes for which an adequate analytical model is not available, pose a challenging task to meet the control objective using most of the developed linear and non-linear methods. However if operator expertise or knowledge base is available for these systems, fuzzy theory provides an adequate solution for control design [4]. For some non linear process the model parameters vary with the time or with uncertain initial conditions. The control of such non linear dynamic processes falls under the domain of adaptive control, where the control law adapts itself with the changing dynamics to meet the control objective. Adaptive control of aircraft due to changing mass during the flight is most promising application of adaptive control [5]. Design of fuzzy logic control (FLC) is based on the set of 'If then' rules forming a rule base. The Multi input single output fuzzy rule base has the following form

$$\text{if } x_1 \text{ is } A_1^i \text{ And ..... And } x_n \text{ is } A_n^i \text{ Then } y \text{ is } B^i \tag{1}$$

Where $x_1, x_2, ..., x_n, y$ are the linguistic variables which takes the linguistic values from the fuzzy sets $A_1, A_2, ..., A_n, B$ and $i = 1, ..., l$ is the number of fuzzy rule. The Fuzzy Rule Inference (FRI) employs fuzzy relation to represent the rule base as given in 1 for mapping the input and output space. Based on FRI, various type of control schemes have been developed but two most famous are Mamdani [6] and model based Takagi Sugeno (TS) [7] type fuzzy control systems. Mamdani (conventional) type also called Type-I fuzzy control systems, the output of each rule is a fuzzy set. This approach is intuitive and well suited to direct human input and has been shown to meet various control objective successfully [8],[9]. In model based TS (Type-III) fuzzy control the consequent part of the rule base (eq.1) is not a fuzzy set but a crisp value or linear combination of input values. This methods presents fuzzy system model to similar classical control model so the conventional control theory of stability analysis and robustness can be utilized to solve complex control problems. Here we present a new type of fuzzy control design approach. It has the following features:

- It has the advantages of both the above mentioned methods. Both the antecedent and consequent parts of the rule base are fuzzy sets, so its intuitive and close to human inputs.
- Fuzzy arithmetic has been employed to bypass the implication and defuzzification steps and to achieve design simplicity and computation efficiency.

- Here we have introduced a new type of membership function called Distending Membership Function (DMF). Its parameters have semantic meanings. Its so flexible that every type of existing membership function can be approximated by tunning the parameters. It has two variants; Symmetric and Asymmetric distending functions and both can be utilized for control system design.
- Our approach can be modeled as dynamic fuzzy model just like TS model. So the conventional model based control theory can be applied for stability analysis and robustness issues.

Distending Function

Distending function denoted by $\left(\delta^{(\lambda)}_{\varepsilon,\nu}(0=x)\right)$ is a continuous soft equality function which is monotonically increasing in the interval $(-\infty,\ 0)$ and monotonically decreasing in the interval $(0,\ +\infty)$ and takes the real values in $[0,\ 1]$. There are two types distending functions; symmetric and asymmetric.

Symmetric Distending Function

The Symmetric distending function shown in fig. 1 is given by

$$\left(\delta^{(\lambda)}_{\varepsilon,\nu}(0=x)\right) = \delta_s(x) = \frac{1}{1+\frac{1-\nu}{\nu}\left|\frac{x}{\varepsilon}\right|^{\lambda}} \tag{2}$$

where λ is called the sharpness factor, ν is the threshold value and ε is the tolerance factor. $\delta_s(x) : \mathbb{R} \to [0,1]$ $\nu, \varepsilon \in (0,1)$ and $\lambda \in (1,+\infty)$. Also $\delta_s(0=0)=1$ and $\delta_s(0=\pm\varepsilon)=\nu$.
The distending function changes its shape from trapezoid to a almost a straight line by changing its parameters as shown in the fig. 1

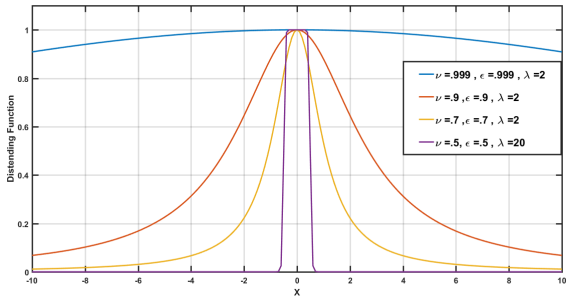


Figure 1: Various shapes of Distending Membership Function

Asymmetric Distending Function

Asymmetric distending function is described by

$$\delta^{(\lambda,\lambda_L,\lambda_R)}_{\varepsilon_L,\varepsilon_R,\nu_R,\nu_L}(x) = \delta_A(x) = \frac{1}{1+\frac{1-\nu_R}{\nu_R}\left(\frac{x}{\varepsilon_R\left(1+e^{-\lambda^* x}\right)}\right)^{\lambda_R}+\frac{1-\nu_L}{\nu_L}\left(\frac{x}{\varepsilon_L\left(1+e^{\lambda^* x}\right)}\right)^{\lambda_L}} \tag{3}$$

where $\delta_A(x) : \mathbb{R} \to [0,1]$ $\nu_R, \nu_L, \varepsilon_R, \varepsilon_L \in (0,1)$ and $\lambda \in (1,+\infty)$. ν_L, ε_L and λ_L are the parameters for the left side whereas ν_R, ε_R and λ_R are the parameters of right side of asymmetric

distending function. Whereas λ controls the overall sharpness of the function and usually its fixed. Asymmetric distending function provides more flexibility in the control design problems as right and left side of asymmetric distending function can be controlled independently and can take different shapes depending on the values of parameters similar to the case of symmetric distending function.

Problem description and proposed design approach

Our design methodology is motivated by work in [10] where a fuzzy control design is approached using the concepts of fuzzy arithmetic. From the operator knowledge base or process data the following rule base can be constructed for a multi-input multi-output system (MIMO)

$$\text{if } x_1 \text{ is } A_1^i \text{ And ..... And } x_n \text{ is } A_n^i \text{ Then } y_1 \text{ is } B_1^i \text{ ; ..... ; } y_m \text{ is } B_m^i \tag{4}$$

Where $x_1, x_2, ..., x_n$ are the input linguistic variables which takes the linguistic values from the input fuzzy subsets $A_1, A_2, ..., A_n$ of universe of discourse $X_1, ..., X_n$. $y_1, y_2, ..., y_m$ are the output linguistic variables which takes the linguistic values from the out fuzzy subsets $B_1, B_2, ..., B_m$ of universe discourse Y. $i = 1, ..., l$ are the number of fuzzy rule and m is the number of the outputs of the system. If the output variables are independent of each other then each rule of the rule base given by eq4 can be written as combination of m multi input single output (MISO) rules of the form

$$\text{if } x_1 \text{ is } A_1^i \text{ And ..... And } x_n \text{ is } A_n^i \text{ Then } y_s \text{ is } B_s^i \tag{5}$$

where $s = 1, ..., m$ are the output of the system. For the sake of simplicity we will consider $s = 1$ and will try to find the fuzzy inference mechanism for mapping the input and output space and generating a crisp output for control signal generation as per rule base. However as stated above it can be generalized to MIMO system if outputs are independent of each other.The antecedent part of the each rule is described by a fuzzy relationship function R_i, given by

$$R_i = A_1^i \cap A_2^i \cap ..... \cap A_n^i \tag{6}$$

where the fuzzy operator $\cap$ is given by the general class of fuzzy operators [11]

$$D_\gamma(x) = \frac{1}{1 + \left(\frac{1}{\gamma} \left(\prod_{i=1}^{n} \left(1 + \gamma \left(\frac{1-x_i}{x_i} \right)^\alpha \right) - 1 \right) \right)^{\frac{1}{\alpha}}} \tag{7}$$

Almost all the existing conjunctive or disjunctive operators (e.g min/max, product, Einstein, Hamacher, dombi) can be obtained from the generalized operator [11] in eq.7. R_i is defined over the Cartesian space $X_1 \times X_2 \times ... \times X_n$. Let the firing strength of ith rule is w_i, i.e.

$$w_i = w_{i1} \cap w_{i2} \cap ..... \cap w_{in} \text{Where } w_{i1} = Poss[A_1^i \mid x_1] \tag{8}$$

Let $W = \sum_{i=1}^{l} w_i$, then the normalized firing strength of ith rule is define as

$$w_{iN} = \frac{w_i}{W} \text{ and } \sum_{i=1}^{l} w_{iN} = 1 \tag{9}$$

By utilizing the concepts of fuzzy arithmetic [10] and the results of previous section that the linear combination of distending functions is also a distending function, we can state the parameters of the resultant fuzzy output distending function as weighted linear combination of

the parameters of l output fuzzy sets . The resultant output distending function has the following form

$$\delta_R(x) = \frac{1}{1 + \frac{1-\nu_R}{\nu_R}\left|\frac{x}{\varepsilon_R}\right|^{\lambda_R}} \quad \text{and} \quad \nu_R = \sum_{i=1}^{l}(\nu_i w_{iN}),\ \lambda_R = \sum_{i=1}^{l}(\lambda_i w_{iN}),\ \varepsilon_R = \sum_{i=1}^{l}(\varepsilon_i w_{iN}) \quad (10)$$

where ν_i, λ_i and ε_i are the parameters of the ith output fuzzy set. Finally the crisp control output will be the mean coordinate of the resultant distending membership function $\delta_R(x)$.

Simulation and Results

A fuzzy controller is designed using the proposed approach to control the level of the tank at the specified height (reference) by opening/closing of the inlet valve. The difference in the measured level l and reference height called error signal is fed to the fuzzy controller. For controlling the level efficiently, rate of change of the level in the tank is also given to the controller as second input. The controller generates control signal for opening and closing of the valve. The input are fuzzified using the distending membership function. Distending membership functions are also defined for the output control i.e. valve opening/closing signal. Dombi conjunctive and disjunctive operators are selected for evaluating the rules. A reference signal for continuously changing the level of water in the tank between 5m and 15m is fed to the system. The fig.2 shows the comparison of response of proposed control vs the response of the conventional type-I TSK based controller.

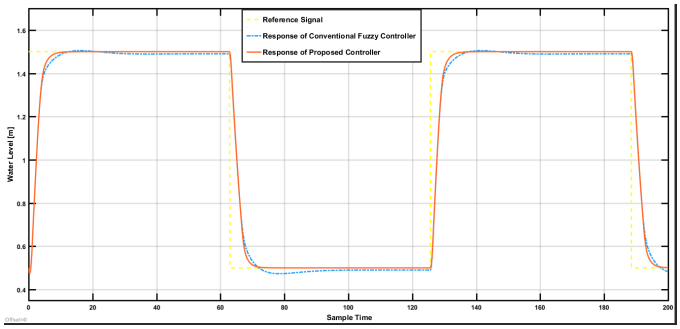


Figure 2: Response Comparison of Proposed and Conventional Fuzzy Controllers

Conclusion and future directions

Fuzzy control based on distending membership function has been proposed. It uses the knowledge rule bases for representing both antecedent and consequent parts and generates the fuzzy model for dynamic processes. Using this model the stability analysis and control design can be achieved. A computationally efficient algorithm has been proposed to design fuzzy controller for dynamic processes. The efficiency of proposed approach has been proved using simulation case study of water tank system . In future, the same design can be extended to design an adaptive fuzzy controller for tunning the membership parameters to overcome the changing process dynamics with increased computational efficiency.

References

[1] L. A. Zadeh, Fuzzy sets, *information and control* , Vol. 8, No. 3, pp.338-353 1965

[2] R. R. Yager, L. A. Zadeh, An introduction to fuzzy logic applications in intelligent systems, *Springer Science & Business Media*, Vol. 165, 2012

[3] L. Yu, Y. Q. Zhang, Evolutionary fuzzy neural networks for hybrid financial prediction, *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, Vol. 35, No. 2, pp. 244-249, 2005

[4] H. Hellendoorn, R.E Precup, A survey on industrial applications of fuzzy control, *Computers in Industry, Elsevier*, Vol. 62, No. 3, pp. 213-226, 2011

[5] A. Karl B. Wittenmark, Adaptive control, *Courier Corporation*, 2013

[6] E. H Mamdani and S. Assilian, An experiment in linguistic synthesis with a fuzzy logic controller, *International journal of man-machine studies, Elsevier*, Vol. 7, No. 1, pp.1-13, 1975

[7] T. Takagi, M. Sugeno, Fuzzy identification of systems and its applications to modeling and control, *Readings in Fuzzy Sets for Intelligent Systems, Elsevier*, pp.387-403, 1993

[8] I. Baturone, F. J. Moreno-Velo, S. Sánchez-Solano, A. Ollero, Automatic design of fuzzy controllers for car-like autonomous robots, *IEEE Transactions on Fuzzy Systems*, pp.447-465, Vol. 12, No. 4, 2004

[9] S. H. Lee, J. T Lim, Multicast ABR service in ATM networks using a fuzzy-logic-based consolidation algorithm, *IEEE Proceedings-Communications*, pp.8-13, Vol. 148, No. 1, 2001

[10] J. Dombi, T. Szepe, Arithmetic-based fuzzy control, *Iranian Journal of Fuzzy Systems*, pp.51–66, Vol. 14, No. 4, 2017

[11] J. Dombi, Towards a general class of operators for fuzzy systems, *IEEE Transactions on Fuzzy Systems*, pp.447–484, Vol. 16, No. 2, 2008

Designing and testing VM allocation algorithms for the CIRCLE Cloud manager

Ádám Belákovics, Arnold Czémán, Imre Szeberényi

Abstract: CIRCLE is an IaaS based cloud manager [1] [2]. It provides an ideal way to effectively use virtualized systems in educational environments. BME (Budapest University of Technology and Economics) has been increasingly utilizing a CIRCLE-based cloud management system for years. However, increasing utilization imposes serious requirements on infrastructure and hence for system administrators. As the number of users increases, lack of service has a major impact on the efficiency of everyday university work. It is indispensable to define and observe the SLAs that ensure the long-term utilization and quality of the system. One such problematic part, regarding the quality of the system's service, is the distribution of physical resources among virtual machines. The problem of assigning virtual machines to physical machines is the main topic of this publication.

Keywords: CIRCLE SLA VM allocation policy CloudSim

Introduction

The CIRCLE cloud system operates with a finite set of servers. Therefore the proper allocation of resources is a serious issue. The system has to fulfil different requirements (lecture, workstation, webserver). To solve this problem, the measurement of the results on different metrics and the accurate knowledge of user needs are essential. Improving the scheduling can be achieved by the creation of an algorithm that enables better results in the Virtual Machine (VM) allocation (Selecting an available Physical Machine (PM) for the VM).

The current scheduler

Allocation of the VMs is performed by the scheduler using a simple priority-based algorithm. The PM with the lowest amount of CPU usage is selected.

- The amount of RAM that is allocated on the machines does not play a role in the selection process. This creates uneven RAM utilization among the PMs. In case of heavy loads the lack of this resource has repeatedly posed a problem in the system. The monitoring system enables queries on historical data. While analyzing the results of the queries, minimal deviation can be observed among the CPU utilization, the opposite is experienced in the case of RAM. Consequently, the algorithm performs well for the CPU utilization but neglects available RAM.

- Bulk launching: Another problem is that if many virtual machines are launched simultaneously, the scheduler assigns too many new VMs to a given PM. The reason for this is that the algorithm makes the decision based on the state of the monitor. However, this information may be out of date because the monitoring system's data is updated in every minute. Solving the problem of bulk launch is especially important because this is one of the most important use case of the system. The lab lessons, which are hosted in CIRCLE, follow this behavior.

Setting up the testing environment

Metrics

The aim of scheduling (in our system) is to provide an equal load on each physical machine. If the resources are allocated equally, the system is able to fulfill the SLA requirements (if the required amount of computing power and RAM does not exceed the available.) Therefore the main metric that we use is the mathematical deviation of the resources between the PMs. The goal of our algorithms is to minimize this value. For resource r the deviation function is the following:

$$D_n^2 = \frac{(p_1^r - \overline{p^r})^2 + (p_2^r - \overline{p^r})^2 + \cdots + (p_n^r - \overline{p^r})^2}{n}$$

where,
p_i^r is the utilization of resource r on the PM identified by i,
$\overline{p^r}$ is the mean value of utilization for a given resource,
n: number of PMs.

In all further reference D^2 will be used marked with d.

Basic algorithms

- **A**(random resource allocation): The PM selected for the new VM is random.
- **B**(CPU based load balancing): The new VM is placed on the PM with the lowest CPU utilization.
- **C**(CPU-RAM based load balancing): The new VM is placed on the PM with the lowest CPU and RAM utilization.
  - **C1**: The decision is based on the following value: $CPUfree(\%) + RAMfree(\%)$.
  - **C2**: The decision is based on the following value: $CPUfree(\%) * RAMfree(\%)$.

The simulation

For the simulation we used a simple python simulator program as we mentioned above, which creates PMs with the specified parameters, then it allocates VMs to these by the specified algorithm. For the reason of its simplicity, its usage is limited to point out the differences between the proposed algorithms. The input is a data set, which resembles our infrastructure. In this infrastructure the resources are heterogeneous. The output is the deviation and other data for the visualization.

Results

The python code generates graphs about the scheduling. This proved useful for understanding the test results. The deviation results are based on an average of several test runs. In all test runs the deviation is calculated after the 100th VM is scheduled.

- **A**(random resource allocation): Average deviation: $d(CPU) = 0.0362, d(RAM) = 0.0597$
- **B**(CPU based load balancing): Average deviation: $d(CPU) = 0.0010, d(RAM) = 0.0490$ For the CPU the deviation is lower by an order of magnitude.
- **C**(CPU-RAM based load balancing): Average deviation for C1: $d(CPU) = 0.0127, d(RAM) = 0.0124$ for C2: $d(CPU) = 0.0132, d(RAM) = 0.0087$

Summary

	A	B	C_1	C_2
d(CPU)	0.0362	0.0010	0.0127	0.0132
d(RAM)	0.0597	0.0490	0.0124	0.0087

- The results for algorithm **B** (which is a current scheduler) indicates the issue introduced above. The RAM deviation is high, nearly as high as for algorithm **A**.

- **C1** and **C2** algorithms perform relatively well with both metrics, by examining the data another problem arises. The average deviation of the PMs allocated resources is constantly increasing. It doesn't differentiate between the values of separate resource usage. Based on algorithm C_1 a VM with a CPU usage of 80% and RAM usage of 20% gets the same rank as one with both values at 50%. This unfortunately makes this algorithm unsuitable for long-term scheduling.

The new algorithm

From the results we can conclude that it is necessary to have an algorithm that makes its decision by two variables but does not allow the long-term divergence seen above.

Prioritized load balancing

C3 (CPU-RAM based prioritized load balancing): The selected machine will be the one with the highest $MIN(CPUfree(\%), RAMfree(\%))$ value. This algorithm will be able to solve the divergence problem. It selects the critical resource and makes its decision by the utilization value.

Correction for heterogeneous systems

If the above algorithm is used in a heterogeneous infrastructure (with varying hardware capabilities), it can make the CPU and RAM utilization rates meaningless. Therefore a correction value is needed. This will be the {CPU, RAM}_weight.

$$PM^{i}_{cpu\_weight} = \frac{PM^{i}_{cpu\_cores}}{\sum_{j=1}^{N} PM^{j}_{cpu\_cores}},$$

where:
$PM^{i}_{cpu\_weight}$: cpu_weight of the PM identified by i,
$PM^{i}_{cpu\_cores}$: number of processing cores of the PM identified by i.

For RAM the same correction value can be calculated.

Prioritized load balancing with weights

D(CPU-RAM based prioritized load balancing with weights): The machine with the highest $MIN(CPUfree(\%) * cpu\_weight, RAMfree(\%) * ram\_weight)$ value will be selected.

Tests

The algorithm D was tested using the previously mentioned python program. The following results can be observed: $d(CPU) = 0.0272$, $d(RAM) = 0.0092$. Based on the deviation values, the algorithm does not performs exceptionally well, the deviation for CPU is of the same magnitude as the algorithm **B** for RAM. On the other hand $d(RAM)$ is much lower, which is a promising result. The reason for these results lies in the characteristics of the input data. In the current input PM data (which reflects the resources of the cloud infrastructure at the university), RAM is the scarcer resource. If we change the characteristics of the input data the algorithm will favor the resource which has the lower available amount for a PM. However this is not a problem, since the algorithm designed to focus on the critical variable. Further examination of the data also shows that the D algorithm does not produce the divergence seen at C_1 and C_2, which makes it suitable for long-term scheduling. This fulfils the requirements.

Future of the research

Although the simple test environment was suitable for the algorithms to be compared, it does not mean that the new algorithm would not have problems. The algorithm was extended to handle bulk launching as well, but our current testing environment is not able to validate its effectiveness. First we need to improve our simulation environment. CloudSim [3] is a software package developed by the University of Melbourne at CLOUDS, which allows the simulation of complex cloud infrastructures. This would allow for more accurate testing and validation of algorithms [4]. In addition, there are several publications about this topic [5] [6], and by examining them we are looking to expand our knowledge, to achieve better results solving the SLA problems of our cloud infrastructure.

Acknowledgements

The research has been supported by the European Union, co-financed by the European Social Fund (EFOP-3.6.2-16-2017-00013).

References

[1] Guba Sándor (2012), Cloud rendszer fejlesztése oktatási környezethez, BME

[2] Guba Sándor (2014), Oktatási felhő kialakítása, BME

[3] Calheiros, R. N., Ranjan, R. , Beloglazov, A. , De Rose, C. A. and Buyya, R. (2011), CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Softw: Pract. Exper., 41: 23-50. doi:10.1002/spe.995

[4] Anton Beloglazov, Jemal Abawajy, Rajkumar Buyya (2012) Energy-aware resource allocation heuristics for efficient management of data centers for Cloud computing, Future Generation Computer Systems, Volume 28, Issue 5, 2012, Pages 755-768, ISSN 0167-739X

[5] Zoltán Ádám Mann (2015), Allocation of Virtual Machines in Cloud Data Centers—A Survey of Problem Models and Optimization Algorithms, Journal ACM Computing Surveys (CSUR) Volume 48 Issue 1, September 2015 Article No. 11

[6] J. Hu, J. Gu, G. Sun, and T. Zhao. (2010) A scheduling strategy on load balancing of virtual machine resources in cloud computing environment. In Parallel Architectures, Algorithms and Programming (PAAP), 2010 Third International Symposium on, pp. 89-96, IEEE

Deep Reinforcement Learning: A study of the CartPole problem

Ádám Budai, Kristóf Csorba

Abstract: One of the major challenges of artificial intelligence is to learn solving tasks which are considered to be challenging for even a human. Reinforcement learning is the most general learning framework among the three main type of learning methods: supervised, unsupervised and reinforcement learning. Most of the problems can easily fit into this framework. Experience shows that a lot of machine learning methods with non-linear function approximators suffers from instability regarding convergence. Reinforcement learning is more prone to diverge due to its ability to change the structure of its training data by modifying the way how it interacts with the environment. In this paper we investigate the divergence issue of DQN on the CartPole problem in terms of the algorithm's parameters. Instead of the usual approach we do not focus on the successful trainings but instead we focus on the dark side where the algorithm fails on such an easy problem like CartPole. The motivation is to gain some further insight into the nature of the divergence issues on a specific problem.

Keywords: reinforcement learning, ALE, CartPole, OpenAI gym, DQN

Introduction

Reinforcement learning (herafter RL) becomes more general by involving interactions into the learning process which makes RL an active learning method. Supervised and unsupervised learning use training data generated independently of the learning algorithm. But an RL agent generates the training data (samples) for itself. Fig. 1 shows the components of the RL framework. The agent is the manifestation of the learning algorithm and it provides an interface to interact with the environment through actions. The environment is described by its state. If we knew the behavior of the environment then we would forecast the next state by knowing the current one. The agent receives a feedback in the form of a reward indicating the success of its action. The objective of reinforcement learning is to find a methodology for choosing actions to gather as many rewards as possible [1, 2].

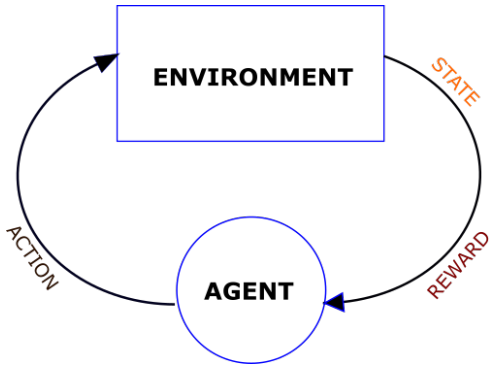


Figure 1: The components of machine learning.

A key concept in the traditional foundation of RL is the so called action-value function (Q):

$$Q(s,a) = E_\tau \left[G(\tau), s_0 = s, a_0 = a \right] \tag{11}$$

$$G(\tau) = \sum_{(s,a)\in\tau} r(s,a,s'). \tag{12}$$

The action-value function shows the value of the current state in terms of the expected return (G) by following trajectories started from s and make action a at first time. If the optimal action-value function is known then the function to choose the next action (policy) is given as:

$$\pi(s) = \arg\max_a Q(s, a). \tag{13}$$

The main purpose in RL is to find the optimal policy. If the optimal action-value function is given then the optimal policy is given by Eq. 13. If the state has high dimension (like an image) then storing the Q function in memory for each state is impossible. Therefore function approximators, like neural networks, are used to represent them. However, training a deep neural network in the RL framework was known to divergent. The DQN algorithm [5] was a breakthrough when it was able to learn a wide range of Atari games [11] from the raw sensory inputs (the image of the playing area) with the same hyper–parameters. After this breakthrough it seems considerable for us that solving the CartPole problem with DQN does not require a lot of parameter tuning but in reality the behavior of the algorithm sensitive for the parameters. In Section the DQN algorithm is introduced. Section introduces the CartPole problem and benchmarking tools to make comparisons with other algorithms. In Section the measured results are summarized after parameter scanning our DQN implementation (available on GitHub [13]). Section discusses conclusion thoughts.

Background

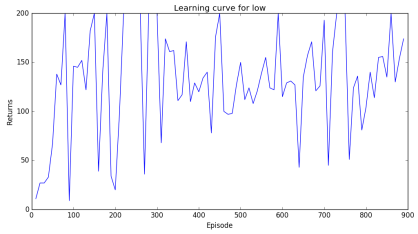
DQN (Deep Q-Network) algorithm [5, 6] is a Q-learning based algorithm where the action-value function (Q-learning) is represented by a neural network. DQN uses two techniques in order to address the issue of convergence. 1) It applies the so called experience replay [8]. It has a significant impact on the learning because the samples (experiences, which are the tuple of state, action, reward and next state, a cycle in Fig. 1) are stored in a buffer. Therefore it can be reusable more then once. This is important in RL where the samples are gather during the interactions with the environment but some part of the environment is difficult to visit, so samples from that part are rare. Throwing them away is wasteful. On the other hand drawing the training samples uniformly from the experience replay to feed into the Q-network breaks the correlations between the samples. This makes the learning process more stable [9]. 2) The so called iterative (or delayed) update uses a copy of the Q-network but it is frozen for a while across bootstrap updates Eq. 14.

$$\Theta_{t+1} = \Theta_t - \alpha \cdot \left(Q(s_t, a_t, \Theta_t) - (r_t + \gamma \max_{a'} Q(s', a', \Theta^-)) \right) \frac{\partial Q(s_t, a_t, \Theta_t)}{\partial \Theta_t}, \tag{14}$$

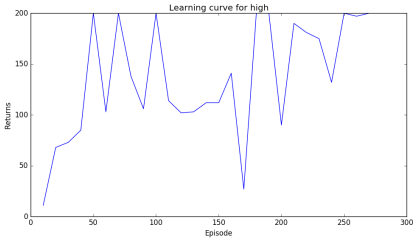
where Θ^- are the frozen weights and it is updated after more steps (in $t + k$ for a well-chosen k), γ is the discounting factor between 0 and 1, r_t is the immediate reward. The usage of experience replay requires an off-policy learning algorithm like Q-learning. Without this the samples should be used sequentially (in the sequence they were sampled from the environment) for updates, otherwise convergence is not ensured even in the case of a linear-approximator. Unfortunately, off-policy algorithms more proned to divergence then on-policy ones. In case of DQN it is ridiculous that experience replay is able to balance this out. The authors claim that in the future developing stable off-policy algorithms is essential for scalable RL which is the base for real life applications. The intuition behind this is that off-policy algorithms make possible to use the same experience (sample) for train different agents parallel. Therefore we investigates DQN. In recent years more versions of DQN were developed [7] but in this paper we implemented the original version because it has many available benchmarks to compare.

Benchmarking tools

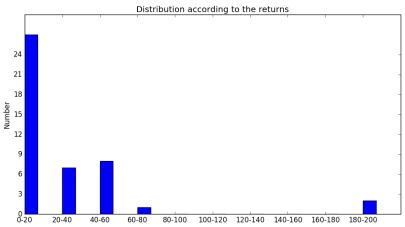
When newer and newer algorithms are developed it is vital to establish testbeds in order to compare the results of new algorithms to older ones. In RL, OpenAI gym is a popular [3] tool. Especially the Atari-games [11] which are based on the Arcade Learning Environment [4]. The Atari games are fairly complicated to solve but the training is time consuming and does not appropriate for parameter scanning. Therefore a simpler problem was necessary which has the same characteristic (the state is an image about the playing area) as the Atari in OpenaAI gym. Classic control problems, like CartPole, is suitable for that. Unfortunately, the original implementation does not support giving back images as states, therefore we changed the implementation in OpenAI gym to get rid of this insufficiency [12]. The CartPole problem: "A pole is attached by an un-actuated joint to a cart, which moves along a frictionless track. The system is controlled by applying a force of $+1$ or -1 to the cart. The pendulum starts upright, and the goal is to prevent it from falling over [10]".



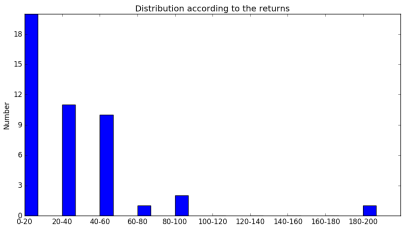
(a) Learning curve for low dimensional input.



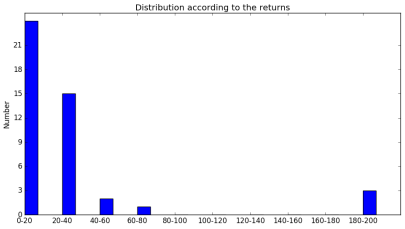
(b) Learning curve for high dimensional input.



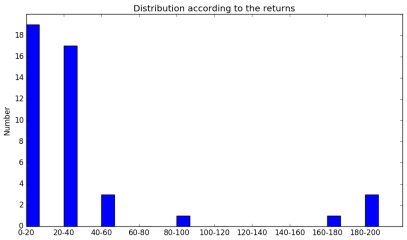
(c) Distrib. of the returns for low dim. input.



(d) Distrib. of the returns for low dim. input.



(e) Distrib. of the returns for high dim. input.



(f) Distrib. of the returns for high dim. input.

Figure 2: The first row shows the learning curves for the low and the high dimensional cases for the best learners. The second and the third row shows the returns achieved by the algorithm on different parameters and organized according to bins with width 20.

Experimental results

Fig. 2 summarizes the results. It can be seen that most of the parameters were wrong to achieve good performance. Among 46 experiments only 2-3 were able to solve the task. The

two columns in the second and third rows are runs of the same algorithm. The parameter table available at [13]. The best parameters for high dimensional inputs is in the 13rd row while for the low case the 14th row in the corresponding files. The corresponding parameters not at the extreme. Deeper, wider networks do not performs better by definition. The result strongly depends on the chosen activation too but as the best algorithms exemplifies [10], both relu and tanh can be successful. Fig. 2 makes the impression that high dimensions are easier to solve. We encourage interested readers to play with the code.

Conclusions

The results show how difficult to find a right parameter settings for this easy problem and the results can vary across executions as well. But regarding the fact that off-policy learning is so important it is worthy to find methods to stabilize it. Furthermore achieving stable, robust results on problems like CartPole can be significant as well.

Acknowledgement

This work was performed in the frame of FIEK_16-1-2016-0007 project, implemented with the support provided from the National Research, Development and Innovation Fund of Hungary, financed under the FIEK_16 funding scheme.

References

[1] R. Sutton, Introduction: The Challenge of Reinforcement Learning, *Machine Learning*, Vol. 8, pages 225–227, 1992.

[2] R. Sutton and A. Barto, Reinforcement Learning: An Introduction, *MIT Press*, 2018

[3] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang and W. Zaremba. OpenAI Gym, *CoRR*, arXiv:1606.01540, 2016.

[4] M. G. Bellemare, Y. Naddaf, J. Veness and M. Bowling. The Arcade Learning Environment: An Evaluation Platform for General Agents, *Journal of Artificial Intelligence Research*, Vol. 47, pages 253-279, 2013.

[5] V. Mnih et. al., Playing Atari with Deep Reinforcement Learning, *CoRR*, Vol. abs/1312.5602, 2013.

[6] V. Mnih et. al., Human-level control through deep reinforcement learning, *Nature*, Vol. 518, doi:10.1038/nature14236, 2015.

[7] Z. Wang et. al., Dueling Network Architectures for Deep Reinforcement Learning, *CoRR*, Vol. abs/1511.06581, 2015.

[8] L. Lin, Self-Improving Reactive Agents Based On Reinforcement Learning, Planning and Teaching, *Machine Learning*, Vol. 8, pages 293–321, 1992.

[9] Y. LeCun et. al., Gradient-Based Learning Applied to Document Recognition, *Proc, of the IEEE*, 1998.

[10] OpenAI gym leader board, https://github.com/openai/gym/wiki/Leaderboard.

[11] Atari games, https://en.wikipedia.org/wiki/Atari_Games.

[12] OpenAI gym with new CartPole environment, https://github.com/adamtiger/gym/tree/openai-goal-based-atari.

[13] The implementation of DQN for the CartPole problem, https://github.com/adamtiger/CartpoleCSCS.

An approximative and semi-automated method to create MPEG-4 compliant human face models

Ákos Tóth, Roland Kunkli

Abstract: In this paper, we introduce our method to facilitate the process of creating an MPEG-4 compliant face model based on a simple 3D mesh. The presented method is semi-automatic, and the user needs to choose 56 points on the model as a preprocessing step. We use a cage based deformation technique to approximate an input model with a generic one which already contains the required MPEG-4 parameters. In the paper, we also show how the cage can be constructed to surround the model completely. The resulting model can be used in any MPEG-4 based facial animation player.

Keywords: cage based deformation, MPEG-4, talking head, facial animation, automation

Introduction

The usage of virtual avatars — also called talking heads in the case of eliminating the body part — is widespread in HCI (Human-Computer Interaction), and they may play an essential role in the future as well.

Methods for creating a talking head based on the face of a real or a fictive person typically consist of two main steps. The first one is the creation of the model itself, while the second step is to prepare our model for further facial animations.

Some of the main challenges are the portability and the reusability of the prepared model. Their principles can be very different; therefore, in most cases, the conversion between the outputs is nearly impossible. For solving this problem, many earlier systems [1, 2] support the well-known MPEG-4 facial and body animation standard [7]. Using MPEG-4, we can guarantee that the desired animation can be attached to any standard model automatically, but unfortunately, the usual creation of an MPEG-4 compliant face model requires a lot of manual user interactions. With all this in mind, our goal in this paper is to give a possible solution to this problem by reducing the amount of required interaction in a semi-automatic way.

In the next section, we discuss some previous works related to the area of MPEG-4 facial animation. We highlight their main advantages and disadvantages as well. In Section 3, we present our method, which can provide a solution to some of these mentioned disadvantages, based on a prepared generic model and a cage based mesh deformation technique. Then, in the last section, we demonstrate the results of our algorithm and our future ideas as well.

Previous work

The face model adaptation (parameterization) heavily depends on the topology of the considered mesh. In the case of simple models, which consist of few hundred vertices, it can be executed easily manually. However, if the models are highly detailed, relatively complex, and include thousands of vertices, then the calibration is not practical. Unfortunately, the adaptation of an existing parameterization to a new facial model is not possible. To solve some of these problems (e.g., the parameterization of the models), Sheng et al. have been released a PDE based method [9] but unfortunately, it does not support the MPEG-4 standard.

MPEG-4 compliant deformation-based methods have been also published. Escher et al. have been proposed a solution [3] which can create standard faces by using a generic model. Lavagetto et al. introduced a method [5] which can help in the calibration process. But due to the limitations of the used deformation algorithm (i.e., RBF and Free Form Deformation [10]), the techniques cannot provide a satisfactory deformation of the generic model, especially in the case of the nose, the eyes, and the chin (see Figure 1).

As we discussed in Section , our primary goals are to ease the adaptation and avoid possible errors caused by the users. We suggest a semi-automatic method based on a generic model which has to be deformed in order to approximate the input one. MPEG-4 parameters are already specified on the generic model. Thus, the face model calibration can be skipped. For the manipulation of the generic

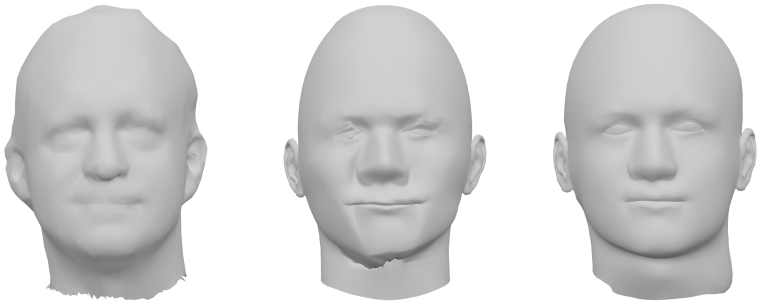


Figure 1: Input model (left), the results of the deformation using Surface-Oriented Free-Form Deformation [10] (middle) and cage based deformation technique (right) which is the basis of our approach.

model, we used a deformation method called cage based deformation technique. One of the main advantages of these methods against other deformation solutions is that using them we can work in real-time, and we have an easy to use, smooth, and intuitive control over the mesh with the defined cage [6].

Our method for automating the face model calibration

As we mentioned in Section , the creation of an MPEG-4 compliant face model requests a lot of time if we do it manually because we need to define all of the 84 feature points and the zone of influence for each FP. Thus, we suggest a semi-automatic solution to create a standard face if its simple 3D mesh is given. At first, we consider a specified generic model together with all its predefined standard points, and then a cage based method is used to approximate the input model with the generic one. Therefore, the whole procedure of the face model adaptation does not have to be executed.

Our deformation method

The necessary modifications of the generic model are achieved by a cage based deformation technique called harmonic coordinates [4]. As we mentioned earlier, a topologically flexible cage (or also known as control mesh) is used to control the deformation of the interior object. If we move the cage vertices C_i to the new positions C_i', an interior point p moves to the new location p', and it is computed as

$$p' = \sum_i h_i(p) C_i',$$

where $h_i(p)$ is the harmonic coordinate of p respect to C_i.

Therefore, at first, we define simple control cages (shown in Figure 2) surrounding the generic and the input models in the same way, based on pre-marked facial feature points on the heads. These cages are responsible for the deformation.

Then, we modify the generic model's cage by translating each point to the corresponding position on the input model's cage separately. As a result, the generic model sufficiently approximates the input one, and it still remains MPEG-4 compliant.

Conclusions and future works

In this paper, we have proposed a robust and semi-automatic method for creating an MPEG-4 compliant face model. Only the definition of the feature points needs user interaction, but it takes just a few minutes. After applying our method, the resulting talking heads can be used in any MPEG-4 compatible facial animation player [8] immediately.

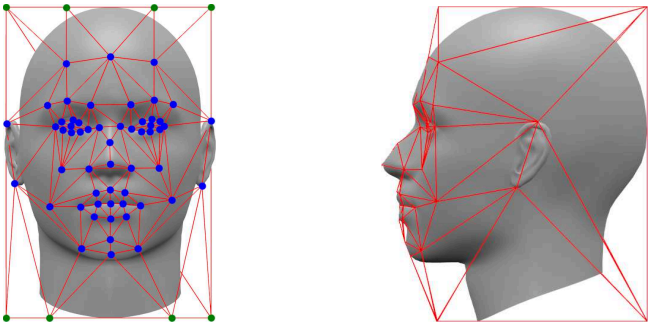


Figure 2: The control cage which has been created for the generic model. Blue dots are the facial feature points of the model, which need to be marked manually, while the green ones are the auxiliary points.

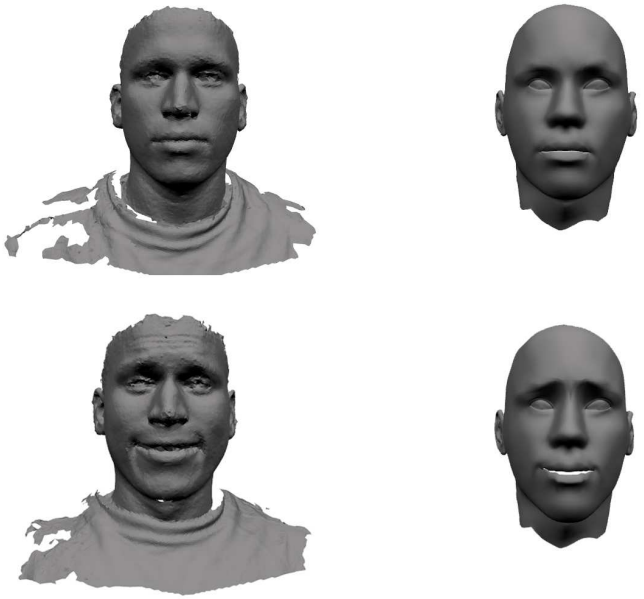


Figure 3: Left: 3D reconstructed human facial expressions from [11]. Right: Our resulting model using the generated FAPs file.

Our solution uses a cage based deformation method called harmonic coordinates to approximate the input model with the generic one. The main difference from earlier systems is that we do not have to define the MPEG-4 parameters. Therefore, calibration errors do not influence the quality of the animation of the model.

To validate our resulting models, we generated FAPs files using 3D reconstructed human facial expressions from the BU-3DFE (Binghamton University 3D Facial Expression) Database [11]. Then these FAPs files were played on our resulting face models to get the same expressions. The original reconstructed model and our deformed model with different facial expressions can be seen in Figure 3.

Additionally, we would like to attempt to minimize the number of user interactions by using a face tracking method which can mark the necessary facial feature positions for us.

Acknowledgement

The first author was supported by the construction EFOP-3.6.3-VEKOP-16-2017-00002. The project was supported by the European Union, co-financed by the European Social Fund.

The second author was supported by the ÚNKP–17–4 New National Excellence Program Of The

Ministry Of Human Capacities.

References

[1] Koray Balci. Xface: MPEG-4 Based Open Source Toolkit for 3D Facial Animation. In *Proceedings of the Working Conference on Advanced Visual Interfaces*, pages 399–402, 2004.

[2] Fiorella De Rosis, Catherine Pelachaud, Isabella Poggi, Valeria Carofiglio, and Berardina De Carolis. From Greta's mind to her face: Modelling the dynamics of affective states in a conversational embodied agent. *International Journal of Human Computer Studies*, 59(1–2):81–118, 2003.

[3] M. Escher, I. Pandzic, and N. M. Thalmann. Facial deformations for MPEG-4. In *Computer Animation 98. Proceedings*, pages 56–62. IEEE, 1998.

[4] Pushkar Joshi, Mark Meyer, Tony DeRose, Brian Green, and Tom Sanocki. Harmonic Coordinates for Character Articulation. *ACM Trans. Graph.*, 26(3):71, 2007.

[5] F. Lavagetto and R. Pockaj. The facial animation engine: toward a high-level interface for the design of MPEG-4 compliant animated faces. *IEEE Transactions on Circuits and Systems for Video Technology*, 9(2):277–289, 1999.

[6] Jesús R. Nieto and Antonio Susín. *Cage Based Deformations: A Survey*, pages 75–99. Springer Netherlands, Dordrecht, 2013.

[7] Igor S. Pandzic and Robert Forchheimer, editors. *MPEG-4 Facial Animation: The Standard, Implementation and Applications*. John Wiley & Sons, Inc., New York, NY, USA, 2003.

[8] Roland Rácz, Ákos Tóth, Ildikó Papp, and Roland Kunkli. Full-body animations and new faces for a WebGL based MPEG-4 avatar. In *CogInfoCom 2015: 6th IEEE International Conference on Cognitive Infocommunications*, pages 419–420, 2015.

[9] Yun Sheng, Phil Willis, Gabriela Gonzalez Castro, and Hassan Ugail. *PDE-Based Facial Animation: Making the Complex Simple*, chapter Poster, pages 723–732. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[10] Karan Singh and Evangelos Kokkevis. Skinning Characters using Surface Oriented Free-Form Deformations. In *Proceedings of the Graphics Interface 2000 Conference*, pages 35–42, 2000.

[11] Lijun Yin, Xiaozhou Wei, Yi Sun, Jun Wang, and M. J. Rosato. A 3D Facial Expression Database for Facial Behavior Research. In *7th International Conference on Automatic Face and Gesture Recognition (FGR06)*, pages 211–216, 2006.

Feature Level Metrics Based on Size and Similarity in Software Product Line Adoption

András Kicsi, Viktor Csuvik

Abstract: Introducing software product lines is a natural way to cope with a large number of software variants and hard maintenance. This task can become more complicated with a fourth generation language, namely Magic in our case. Feature extraction is an important task of product line adoption, and the extracted features can amount to large proportions of the code and can be hard to contemplate, thus appropriate methods become necessary to ease the handling of the information gained. In this work we present some feature level metrics aiming to highlight valuable information on both the results attained through extraction and the features themselves which can be used in furthering the process of product line adoption. We present some metrics based on size and pairwise similarity of the features of four different variants of the same system. The knowledge of these metrics, properly measured and used can be vital in aiding product line adoption.

Keywords: spl, 4gl, feature, magic, metrics

Introduction

Software product lines (SPL) are a common way to cope with the reuse of large software systems. In case of an existing system product line adoption can significantly ease maintenance. SPL adoption is an extensive research branch of software development but literature mostly deals with traditional environments with less emphasis on fourth generation languages (4GL).

Our subject system is a pharmaceutical wholesaler logistics system which was started more than 30 years ago, and is used with great popularity ever since. This popularity has led to the introduction of more than 20 variants of the system which have varying development cycles and isolated maintenance. Our industrial partner is the developer of market leading solutions in the region, which are implemented in the Magic XPA 4GL language [7].

Magic as a language presents some unique challenges. As a fourth generation language development relies heavily on the user interface and produces no source code in the traditional sense. This presents an obstacle for most mainstream static analysis techniques. Magic as a language has also developed a lot in the last 30 years, it had gone through several major version changes introducing significant changes in the structure of the language. There are currently 19 active variants of our subject system written in four different main versions of Magic. A software written in Magic can consist of one or more projects. Projects have their own programs, which are the closest things to the methods of a traditional programming language. Programs can be called and they can call each other too. These programs build up the essence of the functionality provided by the project. We consider these as the building blocks of the features, one program can take part in more features and one feature is usually provided by the work of several programs. Programs can use several data objects through which they access the data stored. Since Magic is a data intensive language, these play a significant part too.

In our previous work [4, 5] we defined our process for feature extraction in the same environment. We presented our work with an information retrieval method based on Latent Semantic Indexing which relies more on the textual similarities located in the features and programs and a technique based on the program calls inside a system, which was based on static analysis and the building of a call graph. We have been provided with a multi-level feature list by domain experts which describes the features present inside the variants of the system. For the sake of clarity we only present the results attained with the highest level of features, of which we distinguish 10 different features. The main contribution of this work is the proposal of several potentially useful metrics on the extracted features of 4GL product line adoption which previously did not exist by our knowledge.

Proposed Metrics

In the current section we define our proposed metrics and display some results attained through their computation on four different variants of the system under study. We mention these variants simply as

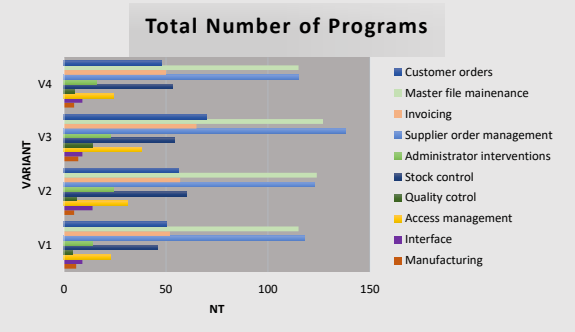


Figure 1: Number of programs by variant

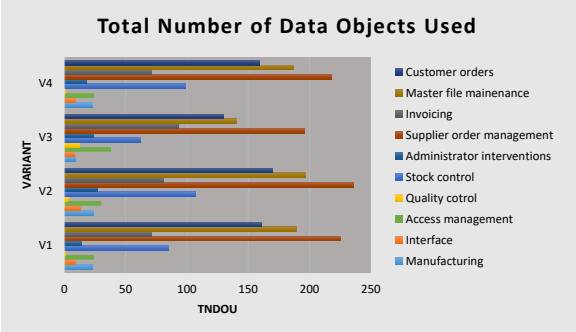


Figure 2: Number of data objects by variant

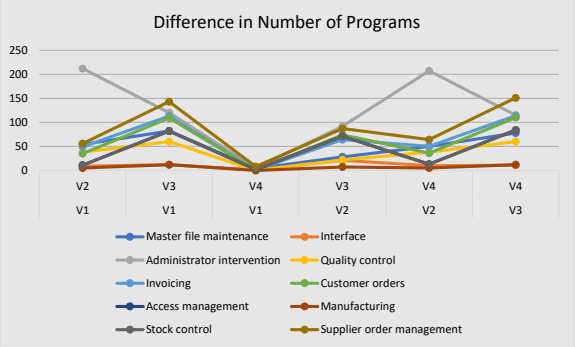


Figure 3: Difference in number of programs at each feature by pairs of variants

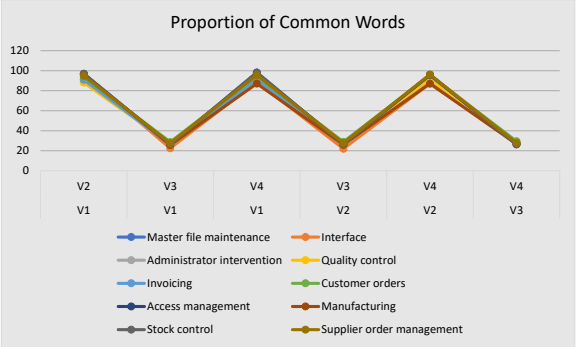


Figure 4: Proportion of common words at each feature by pairs of variants

V1, V2, V3 and V4. According to our previous knowledge V3 differs from these variants highly, while the others share a great number of programs and support a very similar set of functions but still vary somewhat in the specifics. The largest of the variants contains 4251 programs and uses 1065 data objects. Since we have experimented with different feature extraction outputs [5], we have chosen one of these for display, specifically the output achieved with our information retrieval based method which relies on textual similarity attained through Latent Semantic Indexing. The metrics proposed in this section were all measured but due to space limitations we only display the results of just a few of them.

Since at the feature extraction phase we worked on assigning programs to specific features the most straightforward metrics to think of here is the number of programs assigned to each feature. Results of this metric are displayed in Figure 1. On the other hand the number of features assigned to each program can be measured too. These metrics provide basic understanding of the size of features and the importance of the specific programs in their workings. These can be crucially important since we can get a picture of the complications resulting from changes done in a single program. Since a Magic application can have more than one projects, the number of these are also important on the feature level too. In our case the applications always had one single project.

Programs also access data objects to gain the data needed to perform their tasks. Reliance on data objects is another information that can be important for working with these features as sets of their programs. One such metric is the total number of data objects used, for which results can bee seen in Figure 2. In addition to this, since data objects are located in data sources and consist of columns and can even have indices, the number of these are measured too. In our case there are three different data sources, of which most features tend to use all three.

Feature similarity can be measured for each pair of features. Computing these metrics to features of the same variant can also have uses, for example if we are contemplating merging some features on the lower level, but the main importance lays in computing difference between the same feature of two different variants. Similarity can be interpreted in many ways in this case. Since we can handle features as sets of programs, the most straightforward approach is the number of common programs or the absolute difference in the number of programs which is displayed in Figure 3. The average number

of programs or the proportion of size of the smaller feature relative to the larger one can also provide a quick and easy glance at similarity. These metrics are based on feature size and while they can indicate similarity we introduce some textual methods too. One of these is simply the number of unique words contained in their programs which is also based on size but works at a more conceptual level. We can also calculate the proportion of common words which is displayed in Figure 4. It is easy to see from this figure that V3 differs from the rest of the variants greatly which corresponds to our previous knowledge. Textual matching can also be applied as only partial matching when one term appears inside another. We computed both the number and proportion of these sub matches. For these textual metrics textual preprocessing was applied to the text of the programs.

Related Work

Software product line adoption has been an intensively studied subject during recent years [1, 10, 6]. The identification of specific features inside the software is usually one of the major parts of this process [3]. Reverse engineering of 4GL languages, particularly Magic is not extensive, but certain efforts at optimization are visible topics in the Magic community [2].

There already exist several quality measures specifically developed to Magic environments [9, 8]. Our current work deals with the extension of these efforts to the feature aspects of the applications, which can be a valuable asset in product line adoption.

Conclusion

In our current work we present a number of metrics used in attaining more thorough understanding of features extracted from different variants of a Magic 4GL software. In particular, we presented some fundamental metrics based on the size of extracted features and some similarity metrics for the pairwise comparison of features as well. We have computed these metrics on 4 different variants of the application with 10 top level features identified.

These metrics can be computed for each feature extraction output, particularly useful for reaching deeper understanding of the inner workings of the system. With size metrics we can get a good picture of the magnitude of work ahead of us in case of each feature as well as a hint to their complexity. With similarity metrics we can actually see some of the differences between variants and where those differences lay, thus pointing out the specific locations where differences are to be breached, focused on actual functionality.

Acknowledgements

This project was supported by the European Social Fund (EFOP-3.6.3-VEKOP-16-2017-00002).

References

[1] Wesley Klewerton Guez Assunção and Silvia Regina Vergilio. Feature location for software product line migration. In *Proceedings of the 18th International Software Product Line Conference on Companion Volume for Workshops, Demonstrations and Tools - SPLC '14*, pages 52–59, New York, New York, USA, 2014. ACM Press.

[2] John V. Harrison and Wie Ming Lim. Automated Reverse Engineering of Legacy 4GL Information System Applications Using the ITOC Workbench. In *10th International Conference on Advanced Information Systems Engineering*, pages 41–57. Springer-Verlag, 1998.

[3] Evelyn Nicole Haslinger, Roberto E. Lopez-Herrejon, and Alexander Egyed. Reverse Engineering Feature Models from Programs' Feature Sets. In *18th Working Conference on Reverse Engineering*, pages 308–312. IEEE, oct 2011.

[4] András Kicsi, László Vidács, Árpád Beszédes, Ferenc Kocsis, and István Kovács. Information retrieval based feature analysis for product line adoption in 4gl systems. In *Proceedins of the 17th*

International Conference on Computational Science and Its Applications – ICCSA 2017, pages 1–6. IEEE, 2017.

[5] András Kicsi, László Vidács, Viktor Csuvik, Ferenc Horváth, Árpád Beszédes, and Ferenc Kocsis. Supporting product line adoption by combining syntactic and textual feature extraction. In *New Opportunities for Software Reuse : 17th international conference, icsr*. Springer International PU, 2018.

[6] Crescencio Lima, Christina Chavez, and Eduardo Santana de Almeida. Investigating the Recovery of Product Line Architectures: An Approach Proposal. pages 201–207. Springer, Cham, may 2017.

[7] Magic Software Enterprises Ltd. Magic Software Enterprises. http://www.magicsoftware.com, last visited May 2017.

[8] Csaba Nagy, László Vidács, Rudolf Ferenc, Tibor Gyimóthy, Ferenc Kocsis, and István Kovács. MAGISTER: Quality Assurance of Magic Applications for Software Developers and End Users. In *26th IEEE International Conference on Software Maintenance*, pages 1–6. IEEE Computer Society, September 2010.

[9] Csaba Nagy, László Vidács, Rudolf Ferenc, Tibor Gyimóthy, Ferenc Kocsis, and István Kovács. Complexity measures in 4gl environment. In *Computational Science and Its Applications - ICCSA 2011, Lecture Notes in Computer Science*, volume 6786 of *Lecture Notes in Computer Science*, pages 293–309. Springer Berlin / Heidelberg, 2011.

[10] Marco Tulio Valente, Virgilio Borges, and Leonardo Passos. A Semi-Automatic Approach for Extracting Software Product Lines. *IEEE Transactions on Software Engineering*, 38(4):737–754, jul 2012.

Multi-Cloud Management Strategies for Simulating IoT Applications

András Márkus, Attila Kertész

Abstract: Currently, the Internet of Things (IoT) paradigm is closely coupled with Cloud technologies, and the support for managing IoT data is one of the primary concerns of Cloud Computing. In IoT Cloud systems, sensors and different smart devices are connected to the cloud, and large amounts of data are generated by these things that need to be managed by infrastructure Cloud services. Simulation platforms have the advantage of enabling investigations of complex systems without the need of purchasing and installing physical resources. In our previous works, we chose the DISSECT-CF simulator to examine IoT Cloud systems, and developed cost-aware policies for managing IoT and Cloud components. The aim of this paper is to further extend the simulation capabilities of this tool by introducing multi-cloud management and selection approaches. We detail our proposed method for the extension, and evaluate multi-cloud utilization through a meteorological case study.

Keywords: Internet of Things, Cloud computing, Simulation

Introduction

In the paradigm of Internet of Things (IoT), sensors and smart devices are connected to the Internet giving way to many opportunities to use cloud and IoT services together [1]. Since more and more devices enter the network to form IoT systems, the dataflow and the workload of the supporting services are increasing. Hiring physical machines from virtual server parks fitting various IoT scenarios could be very expensive, and the investigation of IoT enabled cloud service compositions is not always possible with real cloud providers. As a result in many cases cloud simulators are applied to address such examinations with adequate results.

While network simulators could be too complex to simulate IoT and cloud systems together, due to detailed network configurations, special purpose cloud simulators may be over-tailored to cloud-specific details making it hard to express IoT needs. The number of IoT devices and usage areas are constantly growing, and some cases require immediate intervention after data processing, such as heart monitoring in smart homes, or traffic control in smart cities. This means we need new solutions and techniques for data storage, access and processing, which can be designed and evaluated in infrastructure cloud simulators extended with IoT simulation capabilities. Therefore we have chosen DISSECT-CF to perform our investigations [2].

In our earlier works we introduced and combined provider pricing schemes with IoT cloud management in DISSECT-CF [3]. Since cloud federations provide better services to users, the next step in our research is to enable the usage of multiple cloud datacenters to serve certain IoT scenarios. Therefore in this paper we introduce multi-cloud definition to DISSECT-CF, and propose three cloud selection strategies to be used for mapping IoT devices to cloud services. Finally, we evaluate our proposal through scenarios derived from a real-life weather forecasting service. The weather stations usually have different sensors and usage frequencies affecting data generating methods that can influence cloud service operation and also provider pricing.

Previous extensions of DISSECT-CF towards IoT

One of our main goals for choosing the DISSECT-CF cloud simulator for our investigations was its unified resource sharing mechanism. In this simulator we have two types of events: (i) recurrent time-dependent events that have a frequency value (e.g. 10 ms) which calls their methods regularly in every moment based on the given value, and (ii) the non-recurrent time-dependent events that have only a delay value (e.g. 5ms) denoting the time to be elapsed before its function has to be called. With these build-in events we can simulate the management of IoT systems including sensors and smart devices. The configuration of IoT system properties, such as network bandwidth, local repository size, operating time, target repository, number of sensors and frequencies can be done with an XML description. Our earlier extension also includes an application to simulate IoT data processing [3].

Considering provider pricing, another two XML descriptions can be used to set cloud side pricing and IoT side pricing. Usually the cloud side pricing is used to calculate the costs of virtual machines (VMs) used to run an IoT application. It defines a fixed monthly cost per VM instance, but some providers charge the hour per price for every instance the IoT application needs. To manage data coming from IoT devices and sensors, we need to calculate the IoT side costs, that can also be set based on real provider pricing schemes (e.g. Amazon, Azure, IBM and Oracle – as defined in [3]). In general, the IoT prices are calculated after the generated data traffic in MB following the "pay as you go" approach, while some providers charge after the number of messages exchanged in a month, or set a monthly device per price or messages sent in a day.

By executing a simulation, the following steps are taken: First, a cloud is set up using an XML description (we used the model of a Hungarian private infrastructure called the LPDS Cloud of MTA SZTAKI), and the necessary amount of stations are initialized, and the VM parameters are loaded from additional XML files, which also describe the cloud and IoT costs. Next, the application is started to deploy an initial VM and to start the metering and data generation processes of device stations, and to start IoT and cloud operation price estimation in parallel. During the application execution, a service checks if the cloud repository received a scenario-specific amount of data, if so, then a compute task will be generated which use the cloud resources for data processing. Finally, data generation by sensors, dataset allocation to virtual machines, compute task execution and possible starting and stopping of virtual machines are repeated till the end of the simulation.

The proposed cloud selection strategies

The main research question of this paper is how we can influence the behavior of an IoT application, if the sensors can have different allocation strategies for multiple clouds. In the earlier version of the extended DISSECT-CF we had only one cloud datacenter to start VMs, therefore all sensors and smart devices was connected to this specific cloud, and all the generated data of the sensors were processed by virtual machines running in the same cloud. This cloud had a preloaded cost calculation policy with a single pricing scheme. As a result, a single cloud could be easily overloaded, and the unprocessed data could hinder the operation of the IoT application causing longer response times, even service unavailability for real-time solutions.

In this work we introduce the possibility of the multi-cloud management for IoT cloud simulations in DISSECT-CF. During the start of the simulation we can set up different clouds using the extended XML description denoting sets of physical machines and repositories with various properties. We can associate different pricing policies to the defined clouds, and within a simulation the application can decide to which cloud the IoT devices should be connected, thus where the generated data should be sent and processed.

In the IoT paradigm the sensors are passive entities of the systems, thus their performance is limited by the operation frequency (i.e, data generating, storing, sending to the cloud), uptime and network connection. Usually large amounts of sensor data are sent from the smart devices to cloud resources for further computation and analysis. Since using these resources cost money, IoT system operators can reduce their expenses by selecting a provider having a suitable pricing scheme.

Within this research we defined three different strategies to perform cloud provider selection (to be done for each sensor start-up), which can be denoted by setting the *strategy* field of the XML description of each sensor. These strategies are the followings: (i) *random*: a sensor (or a set of sensors, i.e. a station) chooses one from all the available clouds randomly; (ii) the *cost-aware* strategy looks for the cheapest available cloud (based on their static pricing properties), thus it compares the prices of the required VM for a given sensor. Its algorithm first orders the cloud by their hourly usage prices, then by the VM instance prices. This solution may be more suitable for IoT applications having relatively small data processing needs. Finally, the last one is (iii) the *runtime-aware* strategy, where the corresponding algorithm ranks the available clouds by a specific value defined by the ratio of the number of already connected sensors and the number of the available physical machines of the given cloud. This is a dynamic strategy taking into account the actual load of the available clouds. Applications having longer data processing needs may prefer this strategy.

Evaluation with a weather forecasting scenario

One of the earliest examples of sensor networks comes from the field of weather prediction, therefore we chose to model the crowdsourced meteorological service of Hungary called *Idokep.hu*. It operates more than 400 stations generating sensor data (including temperature, humidity, barometric pressure, rainfall and wind properties), and the actual weather conditions are refreshed on its website in every 10 minutes.

In the scenario we use to model its operation, all stations have 8 sensors and the message size of the sensors can be set up to 0.05 KBs, and the sensors generate data in every minute. The start-up period of the stations were selected randomly between 0 and 20 min. In order to exemplify the usage of different cloud selection strategies, we defined periodic start-up and shut-down dates for certain stations (e.g. to represent malfunctions or failures). We simulate a whole day of operation (from 0:00 a.m. to 24:00 a.m.), and we start the simulation by setting up 200 stations (at 0:00 a.m.). At 2:00 a.m. we start 100 more, and at 10:00 a.m. 200 more to scale the total number of operated stations up to 500. At 2.00 p.m. we shut down 200 stations to scale down the number of running station to 300 by 10 p.m.

With these station management timings we run four different test cases: (i) all stations run with random strategy, (ii) all stations run with cost-aware strategy, then (iii) all stations run with runtime-aware strategy. Finally, (iv) we mixed the three techniques in the last experiment. For this evaluation we used three different clouds with different cloud-side pricing, but the same IoT-side pricing was set for handling the IoT devices in all test cases. Table 1 shows the detailed configurations.

We executed the formerly defined scenario with the four test cases. As we mentioned before, the IoT side cost is 0.352 Euro for all cases using the IBM Bluemix IoT provider. The results of the experiments can be seen in Figure 2. The so-called timeout parameter denotes how much time it took for the application to terminate (i.e. perform all remaining data processing) after the last station stopped working (at 24:00 a.m.).

Table 1: Detailed multi-cloud configuration for the evaluation

Cloud	Physical machine	Hourly price	VM type
LPDS-1	1 PM - 32 cores, 128 GB RAM 5 PMs - 8 cores, 12 GB RAM	0.297 Euro	8 core, 14 GB RAM Azure Large category
LPDS-2	1 PM - 48 cores, 128 GB RAM 5 PMs - 8 cores, 12 GB RAM	0.039 Euro	1 core, 2 GB RAM Amazon Small category
LPDS-3	1 PM - 64 cores, 128 GB RAM 2 PMs - 8 cores, 12 GB RAM	0.150 Euro	4 core, 2 GB RAM Bluemix Medium category

As we can see the cheapest solution for the current IoT application was the runtime-aware strategy with 41 Euros, utilizing 26 virtual machines. The cost of the cost-aware strategy is very close to the runtime-aware one, but it had a 40 minutes timeout, because of the lower speed provided by the weaker virtual machines. The presented scenario and the extended DISSECT-CF with the mentioned XML description formats are available at [4].

Conclusion

In this paper, we extended the DISSECT-CF simulator with the possibility of utilizing multiple clouds for IoT cloud experiments. We presented three different strategies for mapping IoT devices to cloud resources, and exemplified their utilization through a meteorological scenario. Our future work will address the development of other dynamic cloud selection algorithms using fuzzy methods.

Table 2: Evaluation results of the considered strategies

Strategy	**Random**	**Cost-aware**	**Runtime-aware**	**Mixed**
LPDS-1	26,135	0	21,977	23,166
LPDS-2	14,975	43,563	10,218	10,723
LPDS-3	21,750	0	9,140	14,549
Sum (Euros)	62,86	43,563	41,335	48,438
VMs	51	71	26	47
Timeout (min)	20	40	15	15

Acknowledgements

This work was supported by the Hungarian Government and the European Regional Development Fund under the grant number GINOP-2.3.2-15-2016-00037 ("Internet of Living Things").

References

[1] A. Botta, W. de Donato, V. Persico, and A. Pescape. Integration of Cloud computing and Internet of Things. Future Gener. Comput. Syst. 56, pp. 684-700, 2016.

[2] G. Kecskemeti. DISSECT-CF: A simulator to foster energy-aware scheduling in infrastructure clouds. *Simul. Model. Pract. Theory*, 58P2, 2015.

[3] A. Markus, A. Kertesz, G. Kecskemeti. Cost-aware iot extension of dissect-cf. *Future Internet*, 9(3), 2017.

[4] DISSECT-CF extensions towards IoT. Online: https://github.com/andrasmarkus/dissect-cf/tree/pricing. Accessed in March, 2018.

Identity-Based Cloud Authentication Protocol

Andrea Huszti, Norbert Oláh

Abstract: Cloud computing is a recently developed new technology for complex systems with massive-scale services sharing among many users. One of the most important security objectives is to achieve secure user authentication. If it is breached, confidentiality and integrity of the data may be compromised. We propose an identity-based mutual authentication protocol, where identity verification is carried out by multiple servers applying secret sharing technology on the cloud provider side. Robustness and elasticity of the cloud are ensured by its hierarchical structure.

Keywords: cloud computing, authentication, secret sharing, identity-based cryptography

Introduction

Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. OpenStack is one of the most popular cloud computing software. OpenStack Identity service supports multiple methods of authentication, including user name and password, LDAP, and external authentication methods (i.e. Kerberos). There are fears about these systems due to their centralized structure. Services storing password information for a large number of enterprises in a central database are primary targets for hackers (e.g. Golden Ticket Attack [10,12], OneLogin attack). In the scientific literature generally centralized, one-factor [8,9] or two-factor identity verification protocols [3,4] are proposed. However, the concept of multiple-server model may enhance the security level. Brainard et.al. suggested a two-server approach in [2], where two servers together decide on the correctness of the password submitted for authentication. In [6], a multiple-server authentication protocol is designed, where one-time passwords are shared among the cloud servers. Merkle tree or a hash tree [11] is applied for verifying the correctness of the one-time password.

Boneh and Franklin [1] formalized the notion of Identity-Based Encryption (IBE) using bilinear pairings over elliptic curve groups. In IBE setting, the public key of a user can be any arbitrary string, typically the e-mail address. There is no need for Bob to go to the Certificate Authority to verify the public key of Alice. This way an IBE can greatly simplify certificate management. The authors presented an identity-based authentication for cloud computing in [7], based on the identity-based hierarchical model for cloud computing. There are several servers participating in the identity verification process, for each user authentication one encryption and one signature generation are needed.

We propose an identity-based mutual cloud authentication protocol, where user identity is verified by multiple servers. A user authenticates himself by his password with a help of a smart card capable of cryptographic calculations and storing values securely. For each authentication users should give their passwords, hence end-to-end identity verification is achieved. Users decide about the number of servers participate during authentication. The structure of the servers is hierarchical ensuring robustness and elasticity for the cloud. Due to the bilinear property we accomplish the multiple server key exchange and password registration with a single message transmission. In the authentication phase secret sharing is applied, mutual authentication is achieved only by two modular exponentiations besides the fast hash calculations and xor operations.

The proposed scheme

In this section, we describe our protocol. We differentiate two participants: *Users* (U) ask for services from the cloud service provider consisting of several *cloud servers* (C_i). A cloud server which is chosen randomly proceeds the steps of the user authentication. The protocol is composed of two stages: *Registration* and *Authentication*.

Registration
During registration a bilinear map, two hash functions and other system parameters are set. Secret keys are also exchanged between each user and cloud server. We give the definition of the bilinear map.

Definition 1 (Admissible bilinear map)**.** *Let G_1 and G_2 be two groups of order q for some large prime q. A map $\hat{e} : G_1 \times G_1 \to G_2$ is an admissible bilinear map if satisfies the following properties:*

1. Bilinear*: We say that a map $\hat{e} : G_1 \times G_1 \to G_2$ is bilinear if $\hat{e}(aP, bQ) = \hat{e}(P, Q)^{ab}$ for all $P, Q \in G_1$ and all $a, b \in \mathbb{Z}$.*
2. Non-degenerate*: The map does not send all pairs in $G_1 \times G_1$ to the identity in G_2. Since G_1, G_2 are groups of prime order, if P is a generator of G_1, $e(P, P)$ is a generator of G_2.*
3. Computable*: There is an efficient algorithm to compute $\hat{e}(P, Q)$ for any $P, Q \in G_1$.*

Hash functions $H_1 : \{0;1\}^* \to G_1$, $H : \{0;1\}^* \to \mathbb{Z}_q^*$ and two random values $z, \gamma \in \mathbb{Z}_q^*$ are also chosen. In our identity-based setting let $Q = \gamma P$ denote the public and γ the secret key of the Private Key Generator (PKG). For each C_i a $(PK_i = Y_i = H_1(ID_i), SK_i = \gamma Y_i)$ keypair is generated, where the secret key is generated by PKG and ID_i is a publicly known identity information.

Between each U and C_i secret values D_i are exchanged securely in a way that only a digitally signed message is sent by U on a public channel. We assume the existence of a Bulletin Board ($\beta\beta$) that is publicly readable and can be written by PKG only. A public key exchange parameter $x_i P$ is stored for each C_i on $\beta\beta$. Cloud servers build a binary tree structure, secret key exchange parameter x_i is generated by the child elements, in case of leaf elements by the parent and the sister element. If a server breaks down, it can be replaced, therefore scalability and robustness of the cloud is assured.

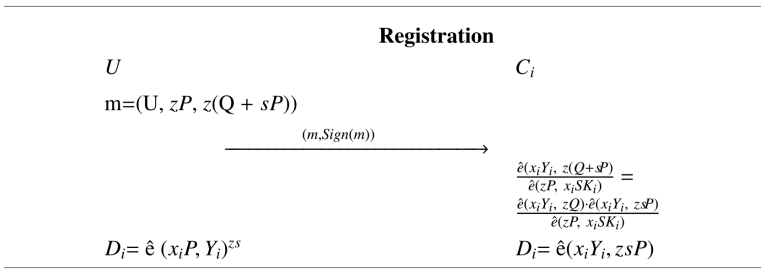


Figure 1: Registration phase

The calculations of the registration can be seen on Fig 1. Each user sends his password pw securely to C_{root}. The message is signed by U with an identity-based signature. C_{root} broadcasts $(m, Sign(m))$, each C_i calculates $D_i = \hat{e}(x_i Y_i, zsP)$. On the other side U computes and stores $\hat{e}(x_i P, Y_i)^z$, by calculating the sth power D_i is generated.

Authentication

In the authentication phase mutual authentication between the user and randomly chosen cloud servers is processed. U chooses k servers $(C_{i_1}, C_{i_2}, \ldots, C_{i_k})$ randomly. Cloud servers C_{i_j}, where $j = 1, \ldots, k$ verify the knowledge of the static password (pw) and the secret key (D_{i_j}) exchanged during Registration.

A random value v is generated by a time-based pseudo random number generator. Cloud server C_v performs the authentication. User U gives his password and his smart card computes $H(D_v^w) = H(\hat{e}(x_v P, Y_v)^{zsw})$. U chooses a bitstring w randomly and generates bitstrings $w_{i_1}, w_{i_2}, \ldots, w_{i_k}$ such that $w_{i_1} \oplus w_{i_2} \oplus \cdots \oplus w_{i_k} = w$, and each share is added to the hashed secret value exchanged during Registration. One can see that the random value w can be calculated on server side only if the cloud servers know the secret values exchanged. Therefore by generating the valid value $H(D_v^{w'}) \oplus w'$ authentication of cloud servers is completed.

Security analysis

Secure mutual authentication prevents adversaries to impersonate a legal user or a cloud server and to achieve illegal access to user data. In this section we give an informal security analysis of the proposed protocol. We consider vulnerability against the replay attack, impersonation attack, and different types of off-line attacks (e.g. dictionary attack, rainbow table attack).

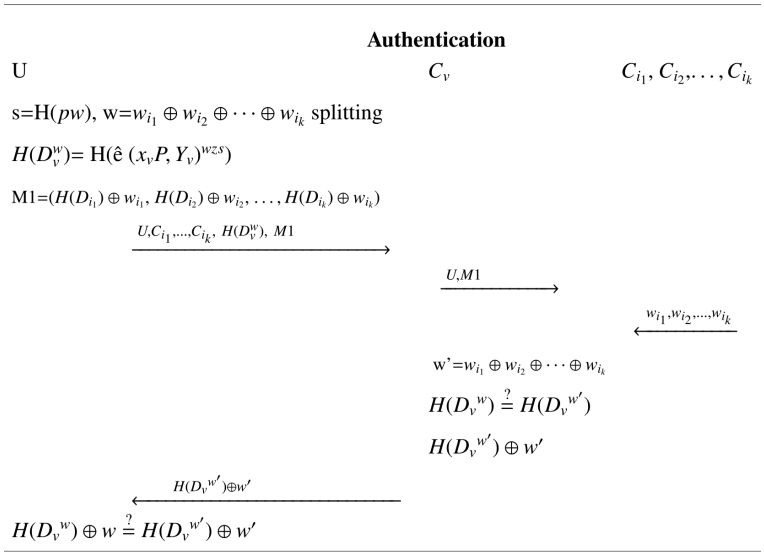


Figure 2: Authentication phase

A replay attack is successful, if by re-transmitting a previously sent message an adversary is able to authenticate himself. Since during authentication each message depends on a randomly generated w, therefore message freshness is achieved providing resistance against replay attack.

In order to impersonate a legal participant secret values D_i are necessary. Values D_i can be calculated only if both the password and secretly stored z, or both the cloud secret key and secret value x_i are known. Secret value x_i is not known by PKG, hence even PKG cannot carry out an impersonation attack. During authentication only the randomized hash values of D_i are sent. Due to the randomness and preimage, collision resistant hash functions no impersonation attack can be performed.

Most of the password-based authentication protocols use verification tables that are vulnerable against off-line attacks. A typical countermeasure is the use of salts. In our system the secret, random value z protects against week passwords.

Since the smart card does not store the password, the user always needs to give the password during the authentication phase, end-to-end identity verification is ensured.

Conclusion

We have designed an authentication protocol for the cloud environment which applies secret sharing and identity-based key exchange. Our system is robust against server breakdown and applies multiple-server identity verification. It is important to note that during the authentication phase we use $k+3$ hash calculations, $3k$ xor operations and two modular exponentiations, hence we have achieved an efficient scheme.

Acknowledgements

This work was supported by EFOP-3.6.2.-16-2017-00015. The project has been supported by the European Union, cofinanced by the European Scofial Fund. The first author is also supported by the Hungarian National Foundation for Scientic Research Grant No. NK 104208.

References

[1] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. SIAM J. Comput., 32(3):586-615, 2003.

[2] J. Brainard, Ari Juels, Burt Kaliski, and Michael Szydlo, A New Two-Server Approach for Authentication with Short Secrets, Proceeding SSYM'03, Proceedings of the 12th conference on USENIX Security Symposium - Volume 12, pp 1-14, 2003.

[3] N. Chen, R. Jiang, Security Analysis and Improvement of User Authentication Framework for Cloud Computing, Journal of Networks, 9(1), pp. 198-203, 2014.

[4] A. J. Choudhury, P. Kumar, M. Sain, A Strong User Authentication Framework for Cloud Computing, Proceedings of IEEE Asia -Pacific Services Computing Conference, pp. 110-115, 2011.

[5] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. Journal of Computer and System Sciences, 28(2):270–299, 1984.

[6] A. Huszti, N. Olah, *A simple authentication scheme for clouds*, Proceedings of IEEE Conference on Communications and Network Security (CNS), (2016), Pages: 565 - 569.

[7] Hongwei Li, Yuanshun Dai, Ling Tian, and Haomiao Yang, Identity-Based Authentication for Cloud Computing, CloudCom 2009, LNCS 5931, pp. 157-166, 2009.

[8] M. S. Hwang, L. H. Li, A new remote user authentication scheme using smart cards, IEEE Transactions on Consumer Electronics, 46(1), pp. 28-30, 2000.

[9] W. C. Ku, S. M. Chen, Weaknesses and improvements of an efficient password based remote user authentication scheme using smart cards, IEEE Transactions on Consumer Electronics, 50(1), pp. 204-207, 2004.

[10] George Kurtz, Dmitri Alperovitch, Elia Zaitsev, *Hacking exposed: Beyond the Malware*, RSA 2015 (slide deck), https://www.rsaconference.com/ writable/presentations/ file_upload/expt10_hackingexposedbeyondthemalware.pdf, (2015).

[11] Ralph C. Merkle, *A Digital Signature Based on a Conventional Encryption Function*, Advances in Cryptology - CRYPTO '87, Lecture Notes in Computer Science, **293**, (1987), pp. 369-378.

[12] Miguel Soria-Machado, Didzis Abolins, Ciprian Boldea, Krzysztof Socha, *Kerberos Golden Ticket Protection, Mitigating Pass-the-Ticket on Active Directory*, CERT-EU Security Whitepaper 2014-007, (2016).

Evaluating the Performance of MQTT Brokers

Biswajeeban Mishra

Abstract: Internet of Things (IoT) is a rapidly growing research field, which has enormous potential to enrich our lives for a smarter and better world. Significant improvements in telemetry technology make it possible to quickly connect things (i.e. different smart devices) that are present at different geographical locations. Telemetry technology helps to monitor and measure the devices from remote locations, making them even more useful and productive at a low cost of management. MQTT (MQ Telemetry Transport) is a lightweight messaging protocol that meets today's smarter communication needs. The protocol is used for machine-to-machine communication and plays a pivotal role in IoT. In cases when the network bandwidth is low or a network has high latency, and for devices having limited processing capabilities and memory, MQTT is able to distribute telemetry information using a publish/subscribe communication pattern. It enables IoT devices to send, or publish information on a topic head to a server (i.e. MQTT broker), then it sends the information out to those clients that have previously subscribed to that topic. This paper investigates the performance of several publicly available brokers and locally deployed brokers by subscription throughput i.e., in how much time a broker pushes a data packet to the client (the subscriber), or how much time a data packet takes to reach the client from the broker, and how the same brokers' performance varies when they are put under stress test? The research question was *"In standard domestic deployment use case, is there any difference in performance of different MQTT broker distributions at standard TCP/IP level?"* MQTT brokers having version v3.1.1 have been evaluated in this study. For the evaluation we use the mqtt-stresser and mqtt-bench stress test tools to evaluate the brokers both locally, and through their publicly deployed brokers.

Keywords: Internet of Things, MQTT, MQTT Brokers, Cloud Computing

Introduction

With the rise of the Internet of Things (IoT), billions of embedded smart devices and sensors are interconnected, and they exchange data using the existing Internet infrastructure. They are enormously impacting and improving our life. In today's fast growing world, many IoT application areas exist, starting from manufacturing, automobile, agriculture, energy management, environmental monitoring to smart cities and the defense sector. For example, a supplying company can trace leakage in oil and gas pipelines from a central control room, and supply can immediately be stopped to avoid accidents [1]. Trace-passing events across the borders of a nation can be detected and sent to the appropriate authorities for necessary action. All these IoT networks use several radio technologies such as RFID (radio-frequency identification), WLAN (wireless local area network), WPAN (wireless personal area network) or WMAN (wireless metropolitan area net-works) to create a Machine-to-Machine (M2M) network. No matter which radio technology is used to operate an M2M network, the end device or machine must make their data available to the Internet. There are many M2M data transfer protocols are available for IoT systems, amongst them, MQTT, CoAP, AMQP, and HTTP are the widely accepted one. Considering message size vs. message overhead, power consumption vs. resource requirement, reliability/QoS vs. interoperability, bandwidth vs. latency, security vs. provisioning, or M2M/IoT usage vs. standardization MQTT, this latest stands tall among all [2]. It is a very lightweight TCP based M2M protocol designed for lightweight publish/subscribe messaging transport. TCP port numbers 8883 and 1883 are registered with IANA for MQTT TLS and non-TLS communication respectively [3]. An MQTT client provides three Quality of service levels for delivering messages to an MQTT Broker and to any client (ranging from 0 to 2). At Qos 0 a message will not be acknowledged by the receiver or stored and delivered by the sender. This is often called "fire and forget." It is the minimal level and guarantees the best delivery effort.At QoS 1 acknowledge is assured. Data loss may occur. At least once delivery is guaranteed. At QoS 2 data delivery is assured. Exactly once delivery is guaranteed. In this paper we investigate the performance of several publicly available brokers and locally deployed brokers , and compare their properties concerning subscription throughput.

Evaluation of Publicly Available Broker Servers

In this section we introduce our evaluation of publicly available MQTT brokers [4], the considered ones are summarized in Table 1. In our test scenario, live environment event data were sent from a Raspberry PI3 Board to the cloud using the MQTT Protocol. So, on Raspberry PI Board side, an MQTT client, called "publisher" was created using a Node-Red programming language to read environment event values from on-board temperature, humidity, and pressure sensors, and publish them on a given topic-tag to an MQTT message broker server at a rate of approximately one message per second. On the receiving end, another MQTT client, called "subscriber" was created to subscribe to the publishing topic, and receive data. Eclipse Foundation recommended MQTT data capture tool MQTT Spy was used to capture, save and analyze the received data. The goal was to evaluate overall topic-specific message load and broker payload of each broker. The evaluation parameters are depicted in Table 2.

Table 1: Publicly hosted MQTT brokers

Type	Mosquitto	HiveMQ	Bevywise
Address	test.mosquitto.org	broker.mqttdashboard.com	mqttserver.com
Port	1883	1883	1883 (TCP)
Sign up Needs	No	No	Yes (name and pwd)

Table 2: Publishing condition parameters for the public experiment

QoS:	0 / 1 / 2
Payload:	14 Bytes
Keep alive time (in seconds)	60
Clean Session	True
Total number of messages published	1000

Evaluation of Locally Deployed Broker Servers

The following MQTT brokers were deployed and evaluated in a local environment with default server configurations: ActiveMQ v5.15.2, Bevywise MQTT Route, HiveMQ 3.3 Evaluation Version, Mosquitto-1.4.14, and RabbitMQ v3.7.2. Concerning the local test environment, all servers (broker and client instances) were deployed in a hardware and software setup shown in Table 3.

Table 3: Hardware and Software Configurations

PROCESSOR:	Intel Core i5-5200U CPU@2.20GHz*4
RAM:	8 GB
DISK:	SATA 3.0, 6.0 Gb/s 5400 rpm
OS:	Ubuntu 16.04.3
OS TYPE:	64-bit
KERNEL VERSION:	4.10.0-42-generic
JAVA VERSION:	Java version "9.0.1"

In the locally deployed MQTT brokers' test scenario, the goal was to evaluate overall message rate and broker payload with a specific test case scenario (i.e: QoS 0/1/2, 1 topics, 1 client). The publishing condition parameters for the locally deployed brokers remained the same as the publishing conditions for public experiment See Table 2; only the message payload was raised to 31 Bytes. A javascript program was created to simulate the sensors in a house. The script simulated and published temperature and humidity sensor values of a room on a unique topic, 1000 times (1 message/second) and thus a total

number of 1000 messages were taken into account to calculate performance statistics of a given broker server.

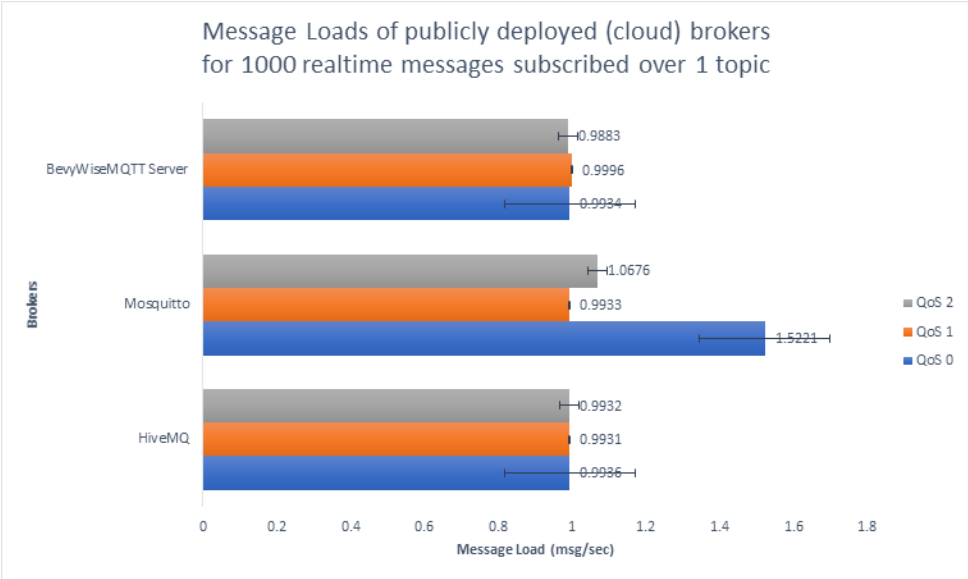


Figure 1: Message Load Rate Public Brokers

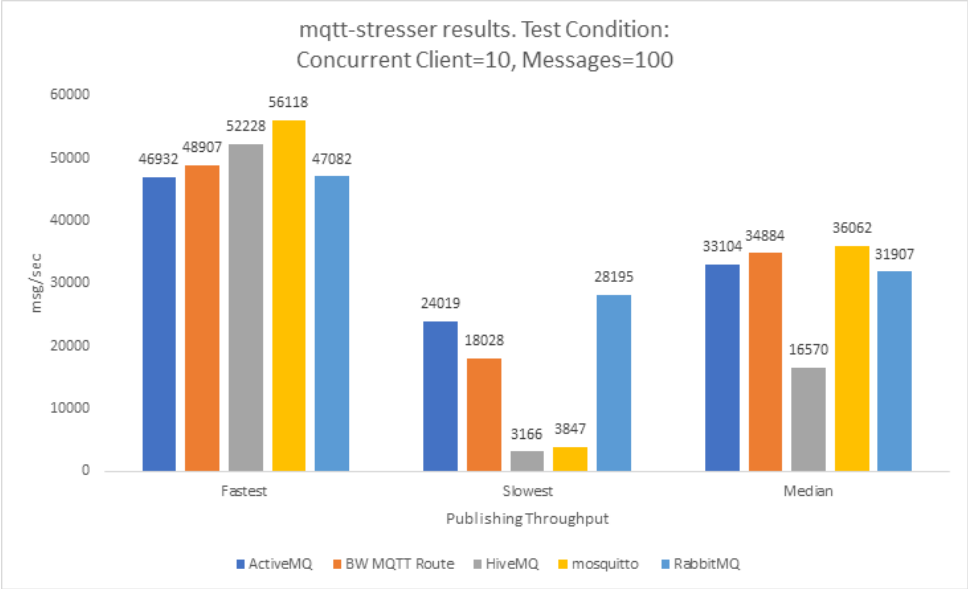


Figure 2: MQTT Stresser Results of Local Brokers

Evaluation results

The evaluation results of the public brokers can be seen in Figure 1, while we depicted the results of the locally deployed brokers in Figure 2 and Figure 3. Based on these experiments, we found that at standard Transport Layer level over TCP/IP there is insignificant difference in the performance of various MQTT brokers in standard domestic deployment use case i.e.: all MQTT broker distributions performed almost identically.

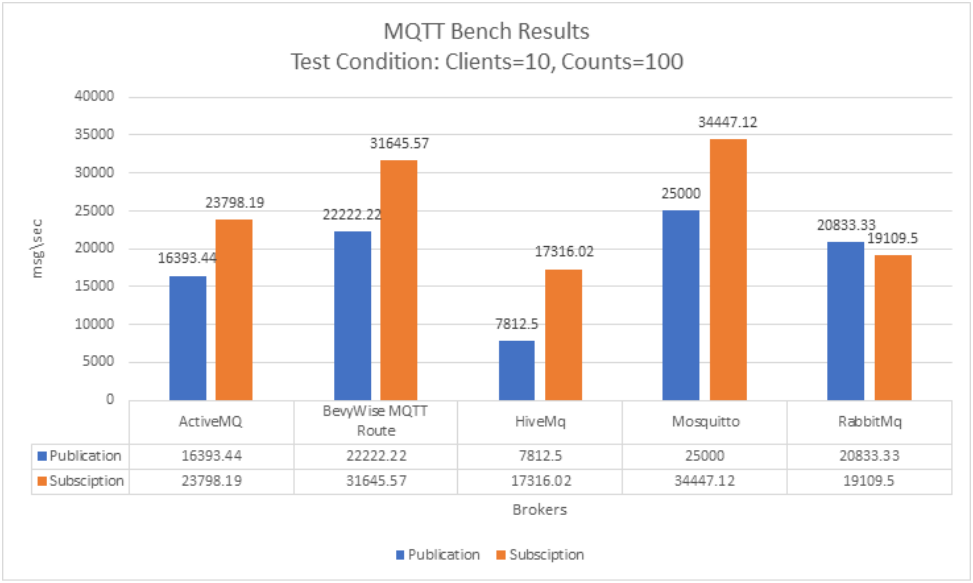


Figure 3: MQTT Bench Results of Local Brokers

Conclusion

The Internet of Things paradigm represents a rapidly growing research field, which has a great potential to ease our lives and to create a better world. There are many M2M data transfer protocols are available for IoT systems, including MQTT, CoAP, AMQP, and HTTP. In this paper we investigated the performance of MQTT brokers and compared their properties by subscription throughput under a specific publishing condition. Our results showed very little difference in the performance of MQTT brokers in standard domestic deployment use case. On the other hand, we found stress testing results of MQTT brokers very inspiring, and we plan to continue this research line in the future with increased number of load conditions.

References

[1] V. Lampkin et al., Building Smarter Planet Solutions with MQTT and IBM WebSphere MQ Telemetry, IBM Redbooks publication, 268 pages, 2012.

[2] Nitin Naik, Choice of effective messaging protocols for IoT systems: MQTT, CoAP, AMQP and HTTP. IEEE International Systems Engineering Symposium, Vienna, pp. 1-7, 2017.

[3] MQTT Version 3.1.1, http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html. Accessed in March, 2018.

[4] Public Brokers, https://github.com/mqtt/mqtt.github.io/wiki/public_brokers. Accessed in March, 2018.

Dynamic business process: comparative models and workflow patterns

Bouafia Khawla, Bálint Molnár

Abstract: TBusiness is rapidly changing nowadays to meet the requirements of a changing environment and to fulfil the customers' needs. All those changes of a business have to be implemented in business processes (BP) and their maintaining information systems (IS). Therefore, a necessity of new approaches for the dynamic business process (DBP) for their modelling, dynamic execution, simulation and management, This gap triggered significant research efforts and a number of approaches to solve DBP modelling problems and ensures the processes dynamicity. In this paper, we present the foundation design of DBP workflow management system (WFMS). We give special emphasis to the investigation of modelling BP, we investigate some fundamental concepts in this domain related with hypergraphs which concepts is strongly supported in Modeling of the BP.

Keywords: DBP, BP modelling, workflow, dynamic environment, hypergraph

Introduction

Modeling of BP [17] is a very active topic in recent years. Because processes are a dominant factor in workflow management, it is important to use an established framework for modeling and analyzing workflow processes [10, 11.12], Modern enterprises and organizations operate in a dynamic environment [4] that is constantly evolving. Organizational theories emphasise that organisation must adapt their environment.

Workflow specifications can be understood, in a broad sense, from several different perspectives [2, 1]. The control-flow perspective (or process) describes activities and their execution ordering through different constructors, The data perspective layers business and processing data on the control perspective. Business documents and other objects which flow between activities, and local variables of the work-flow, The problem will be presented in this research is that the scope of our patterns is limited to static control flow, i.e., we do not consider patterns for resource allocation , case handling , exception handling and transaction management [18]

The organization of this paper is as follows. First, we will start with an introduction to the topic followed by the corresponding literature review, Then we will give workflow definition and various patterns a, then the different tools and languages are defined and the impact of environment when DBP represented , that follows with a simple approach of representation which proposed to mapped DBP in correct way .Finally, we conclude the paper and identify issues for further research.

Literature review

The concept of DBP and DBP modeling [7] is widely differing. Generally, the definitions refer to changes within the external and internal environment[19], and the consequences can be traced through adding, deleting, replacing components representing activities [8]. Optimization algorithm that capable to of continuously adapting a solution to changing environment for each change a different optimization approach required [4]. The combination of internal and external factors that influence a company's operating situation the business environment can include factors such as: clients and suppliers; its competition and owners; improvements in technology; laws and government activities; and social and economic trends,the changes can be accomplished at operational level on the structure of the workflow and instance of a single process, however the modifications of content and business logic included in the activities are not analyzed. Jain et al. [10] examine the impact of internal changes on BPs and the capability of the processes to adapt themselves to the changing environment Hermosillo et al. [18] postulate that processes should be capable for dynamic adaptation to different scenarios, although the method for adaptation cannot be exhibited in detail. Mejia et al. [16] outlines an approach for dynamic adaptation based on Even-Condition-Action.

Workflow

Workflows are a well-established concept to formalize BP and support their execution by a workflow management system. Workflows are "the automation of a BP, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules" [7]. In this section we will present various patterns of which define the basic modeling patterns of BP:

- Sequence: is an ordered series of activities, one activity starting after a previous activity has completed.
- Exclusive choice: defined as check point allows to make a choice of the execution path, the process path is chosen using a decision or condition
- Parallel split: is defined that two or more activities will start at the same time and it is a point in the process where a simple control link is split into multiple control links running in parallel.
- Multiple choice: is differs from exclusive choice pattern in that the multiple-choice pattern allows from one to all the alternative paths may be chosen at performance time
- Multiple merge: is a location in the process where multiple paths merging, but without any control and is a point in the process in which one or more branches of the control thread join without the need for synchronization
- Cycle: is a mechanism repeat the same instructions multiple times with conditions. In next section we will introduce BP modelling tools, their benefits and various kinds and languages .

BP modeling tools

A BP is defined as "a collection of activities that takes one or more kinds of input and creates an output that is of value to the customer" [3]. BP modeling tools allow you to represent your process in a digital way that can then be transferred to a live automated process. There are many benefits to BP modeling:
• Gives a clear understanding of how the process works
• Provides consistency and controls the process
• Identifies and eliminates redundancies and inefficiencies
• Sets a clear starting and ending to the process

There are two kinds of models: Dynamic and Static, most of currently used enterprise modeling technologies can be considered as static. In real life scenarios BPs are not static. That's why demand for non-static models appears. The reference [6,5] contains comparison between static modeling and dynamic modeling: Dynamic model facilitates the display of activities and flow of events within a process. The advantage of using dynamic modeling is that it enables the outcome of a changed process to be evaluated prior to it being implemented into the physical environment. Static models have deterministic nature and are independent of process sequence, it may depend on the data collection and documents that are processed during the flow of information. Currently, most workflow languages support the basic constructs of sequence, iteration, splits (AND and OR) and joins (AND and OR) .However, the interpretation of even these basic constructs is not uniform and it is often unclear how more complex requirements could be supported. Indeed, vendors are afforded the opportunity to recommend implementation level "such as database triggers and application event handling. The result is that neither the current capabilities of workflow languages nor insight into more complex requirements of BP is advanced. Process models are only useful if they actually help to improve processes. For example, verifying the absence of deadlocks in models is a prerequisite for process automation. However, models that are sound but at the same time not used to configure a BPM system do not improve performance. BP need to be managed in environments where processes are only partly documented, and a range of information systems is used. These systems are often "unaware" of the processes in which they are used [9] .

Proposal approach

In this section we will explain main idea of this research, we investigate the opportunities for integration of document-centric approaches for information systems modelling and genre of BP modelling. We have a look at the dynamic aspects of BP, i.e the requirement for changes during operation and the necessary interactions with models of the information system.

We analyse the possibilities for representing the significant artifacts of information systems modelling as processes, the underlying documents and data structures in a unified mathematics representations, namely in hypergraphs which is strongly supported in Modeling of BP . The proposed research gives special emphasis to the investigation of modeling BP, we investigate some fundamental concepts in this domain. The first refers to BP [13] and then BP management [14] Hypergraph is a structure describe complex relationships can be explored among models during analysis and design of IS, it is a generalized graph theory plays a very important role in discrete mathematics [15]

The mappings with hypergraph presented, assigning a formal semantics to workflow languages, together with the "*notionof equivalence,thenallowanin-depthinvestigationintoexpressivenesspropertiesofvarious proprietiesandactivitiesofDBP.*

Conclusion

In this paper, we just propose and present the main idea and describing hypergraph as representation for a DBP, the suggested method takes advantages of the basic properties of generalized hypergraphs. there are some distinguished features: • The structure of the hypergraph can be interpreted as an ontology, thereby it opens the way for DBP representation and reasoning. And considers the time constraints and can be used to analyze the system • It would be a typical specification for a coded solution where we assume that during the coding we find an implementation. • Explain in details in full paper our proposal about DBP modelling and document-centric modelling using the hypergraphs This approach can also be considered as a background to analyse and design another complexes BP or DBP models or complexes IS and shows the hypergraph-based approach offers the chance to apply further mathematical tools for assistance in design.

References

[1] S. Jablonski and C. Bussler. Workflow Management: Modelin]g Concepts, Architecture, and Implementation. InternationalThomsonComputerPress,1996.

[2] W.M.P.vanderAalstandK.M.vanHee. Workflow Management: Models, Methods,and Systems. MIT press, Cambridge, MA, 2002

[3]Hammer, M., Champy, J.: Reengineering the corporation: A manifesto for business revolution. Harper Collins (2009)

[4] Evolutionary optimization in dynamic environments 3 editio book J Branke - 2012 volume 8672 of LNCS, pages 24–39. Springer

[5] Rosenberg. A. (2010) Dynamic versus static modeling types, SAP Modeling Handbook - Modeling Standards, https://wiki.scn.sap.com/wiki/display/ModHandbook/SAP+Modeling+Handbook++Modeling+Standards (accessed: April 27, 2017).

[6] Billington, J., et al. & Weber, M. (2003). The Petri net markup language: concepts, technology, and tools. In International Conference on Application and Theory of Petri Nets (pp. 483-505). Springer Berlin Heidelberg.

[7] Mirjam Minor and Ralph Bergmann and Sebastian Görg.: Case-based adaptation of workflows. Information Systems. 40, 142 – 152. (2014).

[8] Adams M.(2010). Dynamic Workflow, Hofstede et al. Modern BP Automation, Springer-Verlag Berlin Heidelberg,123145

[9] Dumas M (2015) From models to data and back: the journey of the BPM discipline and the tangled road to BPM 2020. In: Proceedings of the 13th International conference on BP management. Springer, Heidelberg

[10] K. Hayes and K. Lavery. Workflow management software: the business opportunity. Technical report, Ovum Ltd, London, 1991.

[11] T.M. Koulopoulos. The Workflow Imperative. Van Nostrand ReinholdNew York, 1995.

[12] P. Lawrence, editor. Workflow Handbook 1997, Workflow Manag ment Coalition. John Wiley and Sons, New York, 1997

[13] Camille Salinesi and Laure-Hélène Thevenet. architecture Enterprise, From practical problems to innovation. Information Systems Engineering, 13(1) :75105, 2008.

[14] Business Process Management: Models, Techniques, and Empirical Studies publié par Wil, van der Aalst, Jörg Desel,Andreas Springer -31 juil 2003

[15] Cui, K,Yang W, Gou, H: Experimental research and finite element analysis on the dynamic characteristics of concrete steel bridges with multi-cracks 19(6).4198-4209 (2017)

[16] Mejía, O., F. Pérez-Miranda, Y. León-Romero, E. Soto-Galera & E. Luna. 2015. Morphometric variation of the Herichthys bartoni (Bean, 1892) species group (Teleostei: Cichlidae): how many species comprise H. labridens(Pellegrin, 1903)?. Neotropical Ichthyology, 13: 61-76.

[17] ARIS, Business Process Modeling August-Wilhelm Scheer Springer Science & Business Media, 27 nov. 2013 - 220

[18] W.M.P. van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski , and A.P. Barros. Workflow Patterns, February 2016, Volume 58(1). 1–6

[19] Bouafia, K, Molnár, B.: Adaptive Case Management and Dynamic BP Modeling A proposal for document-centric and formal approach. In: 12th AIS 2017. (2017).

Binary logistic regression classifying the gender of student towards Computer Learning in European schools

Chaman Verma, Veronika Stoffová, Zoltán Illés, Sanjay Dahiya

Abstract: The authors presented a gender prediction model of student based on answers provided into survey during academic year 2011 in Europe. This experimental study is performed in R-language by applying logistic regression on the large data-set available on the website of European Commission. More than 2500 schools, 27 countries, and more than 45000 students have participated in the survey held in 2011 and survey was conducted by European Commission on primary schools whose were studying at ISCED level 3 (upper secondary level of education). The dichotomous variable is gender and 6 predictors belong to attitude towards computer learning. The best cut-off and accuracy of the presented model is measured 0.499 and 0.628 respectively at 0.5 thresholds using Receiver Operating Characteristics (ROC) and Area under the curve (AUC) which signifies the model to predict more females with correctly as compared to males.

Keywords: Prediction, Confusion Matrix, ROC, AUC.

Introduction

Using the historical data on an attribute or event which can predict the future with specific probability, predictive modeling is required. Logistic regression plays a significant role in many fields which estimate the effects of independent variables on predictor variable as the probability. To modeling binary variables, it estimates relationships of other variables with the dependent variable. In case of various assumption distortions (such as normality, common variances etc.), logistic regression studies and practices are used as an alternative to discriminant analysis and crosstabs. If the dependent variable is binary like 0, 1 or discrete containing more than two levels, as the normality assumption is distorted, it also is an alternative to the linear regression analysis [1]. The term generalized linear models (GLM) usually refers to the large class of conventional linear regression models for a continuous response variable given continuous and/or categorical predictors following [2]. The authors focused on gender variables which are 2-level factor regressing on another 6 variable (related to student's attitude). Logistic regression is suitable in present problem in which gender is categorical and student attitude is numerical. Logistic regression is like discriminant analysis in terms of the aim of estimating a categorical dependent variable, and it necessitates less assumption. On the other hand, if the assumptions necessitated by the discriminant analysis are provided, the logistic regression may also be implemented [3]. Logistic regression was applied to develop the model for the early and reliable prediction of students pass or fail status of the undergraduate level and most found significant factors related to explore pass, fail or drop status of students based on their study [4] [5].The gender is one of the principal determinants of the probability of dropping out. In the binomial probity model, they used, males have a higher probability of dropping out relative to the reference group of females [7].

Experimental Design and Results

An experimental study is conducted in R- language to classify student gender based on their answers given to survey. The responses were belonging to European students of primary schools whose were studying at ISCED level 3 (upper secondary level of education). The dataset consists of more than approximately 150 attributes and 47000 instances. After self-reduction and liwise removal methods, we considered only 8 attributes, 45929 observations from dataset having 19771 male and 26158 female students [1]. An equation of logistic regression is defined to find the probability of accepting female is:

$$P = \frac{e^y}{1 + e^y} \tag{15}$$

where, y is gender.
The gender is considered as class or target variable and questions are predictors and the authors calculated the inter-rater reliability (IRR) of responses dividing the total no. of matched responses (25566)

by total no. of responses (45929).The predictors of model encoded from ST17Q01, ST17Q02 to ST17Q08 based on European Commission survey. To achieve the better performance of the model, dataset is trained and tested randomly at various training ratio of (test, train) such as (0.9,0.1), (0.8,0.2) up to (0.1,0.9). After applying regression model we found six significant variables which have provided meaningful contribution into the study such as ST17Q01, ST17Q02, ST17Q03, ST17Q04, ST17Q07 and ST17Q08 which contributed more than 99%. Hence, we considered only six significant variables into our classification model. To present a significant gender classification model we test and train the dataset using logistic model with 0.5 thresholds, The model equaiton of R- language is given below:

model<glm(GENDER~ST17Q01 + ST17Q02 + ST17Q03 + ST17Q04 + ST17Q07 + ST17Q08, data=train, family='binomial').

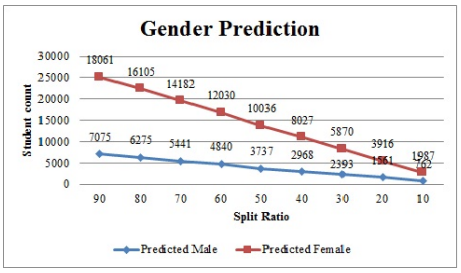


Figure 1: Prediction at Training Ratio (Source: Authors).

Fig.1 reflects gender-wise prediction count of students at the various splitting ratio. It can be seen at the ratio (90, 10), the model predicts more female as compared to male and found to be much significant at this level. The decreasing level of splitting ratio of training data causes poor prediction of gender. At ratio (10, 90), only 762 male and 1987 female are predicted. The gender classification model gives better accuracy at (0.9, 0.1) split. Table 1 shows the total number of the predicted male is 12562 and the total

Table 1: Confusion Matrix (Source: Authors)

Gender	Male (1)	Female (2)
Male (1)	7075	5487
Female (2)	10749	18061

number of predicted female is 28810. Out of total 12562 predicted male, 7075 are predicted accurately and 10749 went to the wrong class named female. It can also see that 18061 students who are predicted as female category correctly with the 6 predictors, while 5487 are predicted into male category incorrectly. The sensitivity or true positive rate (Tpr) of the model is calculated by dividing the total number correct predicted female by the total number of an actual female which is 0.766. The false positive rate (Fpr) is 0.603 which is calculated by dividing the total number of incorrectly predicted female by the total number of an actual male. Further, specificity (1-Fpr) of the model is 0.396 (total number incorrect predicted male/actual male). The precision value is estimated as 0.626 (total no. of correct predicted female/ total no. of an actual female. Hence, at 0.50 thresholds, the accuracy of predicting gender is found 62%. The overall accuracy of the model is calculated as 0.607((total no. of corrected female |+| total no. of corrected male) |/| total no. of students). The prevalence of model is 0.569 which is calculated by dividing the total no. of an actual female by total no. of students which specifies the correct prediction of the female student.

Fig. 2 shows probability is varying in between the range of 0.2 to 0.8. It can be observed that most of the probability to predict the gender of the student against their responses is lies in between 20% to 80%. At the point 0.62 maximum student are found to belongs to female category due to the threshold equation pred1<-ifelse(p1>0.5,2,1) specifying prediction of the female student at 0.5 cutoffs.

The performance of the model is evaluated by ROC and AUC in R-Language. In Fig. 3 the 45 degree reference line (in black) is the line of non-discrimination or benchmark of the model and area under

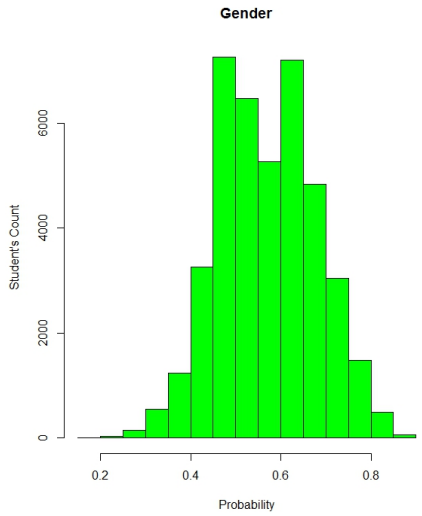


Figure 2: Probability Count (Source: Authors).

the curve (AUC) is 0.6287 which is less than a straight line and ROC curve of the model is above than benchmark or reference line. As the calculated true positive rate (Tpr) or sensitivity is 0.766 and false positive rate (Fpr) is 0.603. The sensitivity of the model measures the ability correct prediction of female and specificity measures the ability to correctly predict the male.

Conclusion

The classification model predicts the student's gender according to their given responses against 6 signficant questions. This model predicts student's gender with more than 60% accurately based on their response. The present study explores the higher count of prediction of female students as compared to male students according to responses and the precision value of the model is measured 0.626. The true positive rate (sensitivity) is 0.766 of model specifies better prediction of the female towards their responses. More than 50% positive predictive value concludes the male and female classes are perfectly balanced. The presented model is efficient to predict student's gender with supporting by ROC curve and confusion matrix.response to computer learning. In future, the sensitivity of model can be enhancing by exploring other remaining variables/features in the survey using other classifiers.

Acknowledgments

The present study is supported by Eotvos Lorand University and Tempus public Foundation, Hungary.

References

[1] European Commission: "https://ec.europa.eu/digital-single-market/news/ict-education-essie-survey-smart-20100039", Accessed on 14 Feb 2018.

[2] McCullagh P., et.al. (1989), "Generalized Linear Models", Second Edition, London: CRC Press Publishers.

[3] Korkmaz1 M., et.al. (2012), "The importance of logistic regression implementations in The Turkish Livestock Sector and logistic regression implementations/fields", J.Agric. Fac. HR.U., 2012, 16(2): 25-36.

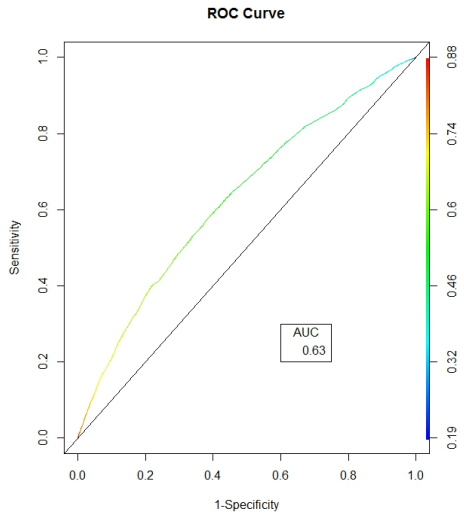


Figure 3: Model Performance (Source: Authors)

[4] Gerard J.A.Baarsa, et.al. (2017), "A Model to Predict Student Failure In The First Year of The Undergraduate Medical Curriculum", Health Professions Education,5–14.

[5] Woodman R. (2001), "Investigation of factors that influence student retention and success rate on Open University courses in the East Anglia region", M.Sc. Dissertation, Sheffield Hallam University, UK.

[6] Boero G., et.al, (2005), "An econometric analysis of student withdrawal and progression in post-reform Italian universities", Centro Ricerche Economiche Nord Sud, CRENoS Working Paper 2005/04.

Operations on Signed Distance Functions

Csaba Bálint, Gábor Valasek

Abstract: We present a theoretical overview of signed distance functions and analyze how sphere tracing algorithms slow down in the presence of set operations on said implicit representations.

Introduction and Previous Work

Surface representations for real-time graphics rely on linear approximations. With the advent of hardware accelerated tessellation units, parametric surfaces gained momentum in real-time computer graphics; however, implicit mappings are still considered infeasible for high-performance applications.

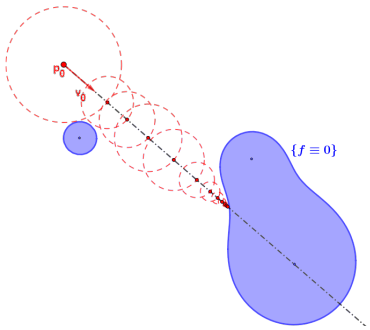
Nevertheless, implicit functions simplify some otherwise difficult operations. For example, the result of set operations on surfaces can be trivially represented by minimum, maximum, and negation of their implicit functions.

Our paper focuses on a particular class of implicit representations, signed distance functions, and their lower bounds. It has been noted by Hart in [1] that they can be rendered efficiently using a technique called sphere tracing.

We discuss this class of functions and highlight their theoretical aspects that have practical consequences in rendering.

Signed Distance Function

From any sample point, a distance function $f : \mathbb{R}^3 \to \mathbb{R}$ evaluates to the distance to the surface $\{f \equiv 0\} := \{\boldsymbol{x} \ : \ f(\boldsymbol{x}) = 0\}$. That is, for any $\boldsymbol{p} \in \mathbb{R}^3$ point in space, the sphere of radius $f(\boldsymbol{p})$ around it is disjoint from the $\{f \equiv 0\}$ surface. This property demonstrates that the sphere tracing algorithm from Figure 1 can be applied to find the first ray-surface intersection.



(a) *Unbounding spheres*

In : $\boldsymbol{p}, \boldsymbol{v} \in \mathbb{R}^3, |\boldsymbol{v}| = 1$ ray; SDFE $f : \mathbb{R}^3 \to \mathbb{R}$
Out: $t \in [0, +\infty)$ distance traveled along the ray
$t := 0; \quad i := 0;$
for $i < i_{max}$ ***and*** $f(\boldsymbol{p} + t \cdot \boldsymbol{v}) > \epsilon$ **do**
  $t := t + f(\boldsymbol{p} + t \cdot \boldsymbol{v});$
  $i := i + 1;$
end

(b) *Sphere tracing adapted from [1]*

Figure 1: *An overview of the sphere tracing root-finding algorithm*

Singed distance functions (SDFs) can represent an entire volume by classifying the points of $\mathbb{R}^3$ belonging to its "interior" ($\{f < 0\}$), "exterior" ($\{f > 0\}$), or to the surface ($\{f \equiv 0\}$). Formally, the continuous function $f : \mathbb{R}^3 \to \mathbb{R}$ is an SDF if $|f(\boldsymbol{p})| = d(\boldsymbol{p}, \{f \equiv 0\})$ for any point $\boldsymbol{p} \in \mathbb{R}^3$. Note that a distance function is also a signed distance function. In summary, we have the following equivalent definition to describe SDFs:

Theorem 1 (SDF equivalence)**.** *The function $f : \mathbb{R}^3 \to \mathbb{R}$ is a singed distance function if, and only if there exists a $\emptyset \neq D \subseteq \mathbb{R}^3$ set for which*

$$f(\boldsymbol{p}) = \begin{cases} d(\boldsymbol{p}, \partial D) & ha\ \boldsymbol{p} \notin D \\ -d(\boldsymbol{p}, \partial D) & ha\ \boldsymbol{p} \in D \end{cases}, \tag{1}$$

where $\partial D := \{f \equiv 0\}$ is the boundary of volume D.

A useful operation on SDFs is offsetting them by a $c \in \mathbb{R}$ distance. Intuition suggests that the function $f - c$ is the SDF of the $\{f \equiv c\}$ offset surface; however, this is only true in a subspace of $\mathbb{R}^3$, as stated by the following

Theorem 2 (Offset of an SDF)**.** *If f is an SDF, then for any $0 \neq c \in \mathbb{R}$ offset, the function $f - c$ is an SDF on the set $\left\{\frac{f}{c} \geq 0\right\}$.*

Signed Distance Function Estimate

Even though formulating SDF representations for arbitrary surfaces is infeasible, lower bounds of SDFs can be readily constructed. Replacing exact distances with conservative estimates retains the correctness of sphere tracing. Mathematically, we define SDF estimates (SDFEs) as

Definition 1 (SDFE)**.** The function $f : \mathbb{R}^3 \to \mathbb{R}$ is a *signed distance function estimate* if there exists a constant $K \in [1, +\infty]$ and a function $q : \mathbb{R}^3 \to [1, K]$ such that the function $f \cdot q$ is a signed distance function.

If the definition holds, we say that f *is a signed distance function estimate with bound* K. The value of q at a given point represents the ratio between the actual and the estimate unbounding sphere radii. Thus, K measures the maximum slowdown of sphere tracing on an SDFE compared to an exact SDF. Note that $K = +\infty$ is allowed. This leads us to the following equivalent definition.

Theorem 3 (SDFE equivalence)**.** *The function $f : \mathbb{R}^3 \to \mathbb{R}$ is an SDFE with bound K if, and only if* $\operatorname{sgn} \circ f$ *is continuous on the $\{f \not\equiv 0\}$ set, and*

$$\frac{1}{K} \cdot d(\boldsymbol{p}, \{f \equiv 0\}) \;\leq\; |f(\boldsymbol{p})| \;\leq\; d(\boldsymbol{p}, \{f \equiv 0\}) \,.$$

The SDFE f is continuous on the surface $\{f \equiv 0\}$, but neither f nor q has to be continuous anywhere else, providing a robust definition of the estimate.

Remark. Often, the surface is given by an arbitrary implicit function, from which one has to generate an SDFE to visualize the surface efficiently. This can be done for Lipschitz-continuous [2] implicit function g by dividing it by its Lipschitz-constant $\operatorname{Lip} g = \inf\left\{L \in \mathbb{R} : \forall\, x, y \in \mathbb{R}^3 : |f(x) - f(y)| \leq L \cdot \|x - y\|_2\right\}$. The function $f := \frac{g}{\operatorname{Lip} g} : \mathbb{R}^3 \to \mathbb{R}$ is an SDFE with a possibly infinite bound.

Lipschitz-continuity on $\mathbb{R}^3$ is a very strong requirement. In most cases, g is Lipschitz-continuous on any compact subset of $\mathbb{R}^3$, like algebraic surfaces. We can overcome this limitation by finding the sphere's radius r for every $\boldsymbol{p}$ center for which the computed SDFE approximation is precisely this r. Thus, for every $\boldsymbol{p}$ point, we have to solve the following fixpoint equation [3]:

$$f(\boldsymbol{p}) := \quad r = \frac{g(\boldsymbol{p})}{\operatorname{Lip} g|_{S_r(\boldsymbol{p})}} \qquad (\boldsymbol{p} \in \mathbb{R}^3) \,.$$

The $S_r(\boldsymbol{p}) = \{\boldsymbol{x} : d(\boldsymbol{x}, \boldsymbol{p}) \leq r\}$ denotes the sphere with center $\boldsymbol{p}$ and radius r.

Set-theorem for Signed Distance Functions

Expanding on Hart's results on set-theoretic operations on objects represented by SDFEs [1], we investigated the quantitative slowdown of sphere tracing after such operations. Using the de Morgan identities, our theorem on intersections can be extended to unions and differences.

Theorem 4 (Intersection I. – Interior)**.** *If f and g are SDFEs with $K_f, K_g \in [1, +\infty]$ bounds, then $h := \max(f, g)$ is an SDFE of the object $D := \{f \leq 0\} \cap \{g \leq 0\}$ with the bound $K = \max(K_f, K_g)$ inside D.*

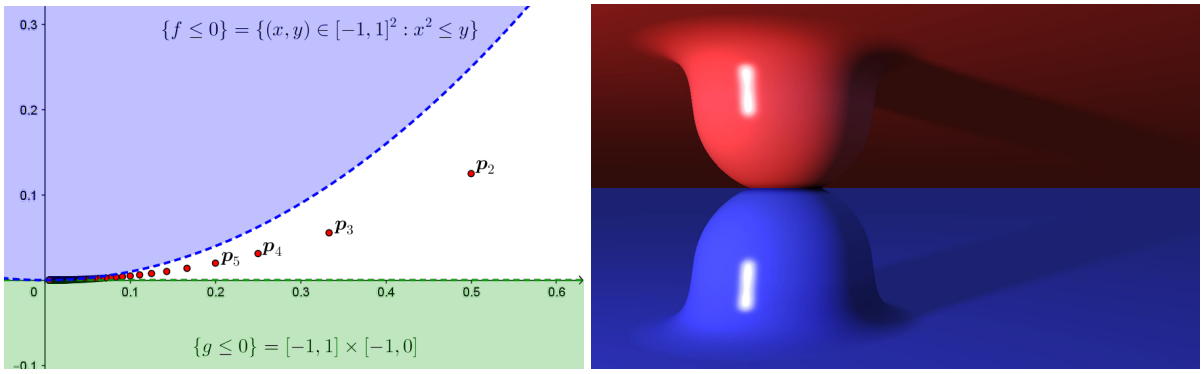
The above means that within the intersection, the resulting SDFE h keeps the precision of the estimate. For the union operation, the exterior subspace of the union benefits from superior performance.

Theorem 5 (Intersection II. – Exterior)**.** *If f and g are SDFs with finite bounds $K_f, K_g \in [1, +\infty)$, and $D = \{h \leq 0\} \neq \emptyset$, and $\{f \leq 0\}$ is a bounded set, then for every $\delta > 0$ on the set $\{\boldsymbol{p} \in \mathbb{R}^3 \;:\; d(\boldsymbol{p}, D) \geq \delta\}$ the SDFE $h = \max(f, g)$ has the finite bound*

$$K_\delta := 2 \cdot \max\left(K_f, \frac{2 \cdot \max(K_f, K_g) \cdot \operatorname{diam}\{f \leq 0\}}{\min(\delta, 2 \cdot d(A_\delta, B_\delta))}\right) \;<\; +\infty \,.$$

Generally, the convergence speed of the sphere tracing algorithm depends on the δ near-cutoff distance, on the smaller objects diameter ($\operatorname{diam}\{f \leq 0\} = \sup\left\{d(\boldsymbol{x}, \boldsymbol{y}) : \boldsymbol{x}, \boldsymbol{y} \in \{f \leq 0\}\right\}$), and its SDFE bound K_f.

If the near-cutoff distance is zero, the theorem does not hold as illustrated by a counterexample on Figure 2a. The $\boldsymbol{p}_n$ sequence is $O(n^{-2})$ close to the two surfaces, but only $O(n^{-1})$ close to their intersection D resulting in $K = +\infty$.



(a) *If $\delta = 0$, then $\boldsymbol{p}_n$ converges faster to $\{f \leq 0\}$ than to $D = \{(0, 0)\}$.*

(b) *$\{f \leq 0\}$ and $\{g \leq 0\}$ are unbounded but their intersection is bounded.*

Figure 2: *Counterexamples explaining the conditions in Theorem 5.*

If none of the interiors $\{f \leq 0\}$ (red) and $\{g \leq 0\}$ (blue) are finite, then a counterexample on Figure 2b disproves the theorem. A point from their small, bounded intersection can be arbitrarily distant while remaining halfway between the two infinite planes, thus also resulting in a $K = +\infty$ bound.

Conclusion

This paper focused on theoretical properties that have a practical effect on rendering. We presented a conceptual overview of signed distance functions and their lower bounds.

We highlighted how offsetting related to precise SDFs and showed a quantitative theorem on how SDF estimates can be used to express the result of set theoretic operations with quantitative bounds.

Acknowledgments The project has been supported by the European Union, co-financed by the European Social Fund (EFOP-3.6.3-VEKOP-16-2017-00001), and supported by the ÚNKP-17-1, New National Excellence Program of the Ministry of Human Capacities, Hungary.

References

[1] John C. Hart. *Sphere tracing: A geometric method for the antialiased ray tracing of implicit surfaces. The Visual Computer*, 12:527–545, 1994.

[2] Kenneth Eriksson, Donald Estep, and Claes Johnson, *Applied Mathematics: Body and Soul*, Number 978-3-540-00890-3. Springer-Verlag 2004.

[3] Csaba Bálint, Gábor Valasek *Interactive Rendering Framework for Distance Function Representations* International Conference on Applied Informatics, 2017

Keeping P4 switches fast and fault-free through automatic verification

Dániel Lukács, Gergely Pongrácz, Máté Tejfel

Abstract: The SDN dataplane is going through a paradigm shift, as softwarization of switches sees an increased pull from the market. Yet, software tooling to support development with these new technologies is still in its infancy.

In this work, we introduce a framework for verifying data plane protocols defined in the P4 language. Using symbolic execution, the framework checks *crash-freedom* and *bounded execution* properties of P4 protocols, and verifies performance requirements by estimating *lower and upper bounds* of packet processing time. This paper explains related terminology, and briefly describes the methodologies used to reach this goal.

Keywords: programmable switches, P4, SDN, performance modeling, symbolic, execution

Introduction

Currently in the networking industry, network devices are being commoditised fast and software gets more and more market share, as consumers want scalable and easily replaceable devices, while vendors want to keep development costs low. Software-defined networking (SDN) [1] technologies address this need by allowing network administrators to dynamically control network topology, configurations, and protocols.

New languages are emerging, aiming to assist network engineers to define the functioning of switches in the SDN forwarding plane. Among them, P4 intends to keep the best aspects of both hardware and software by enabling network engineers to communicate their intent in a general high-level language, while the task of compiling high-level protocol description to low-level target architectures is delegated to the backend software. The hybrid approach is highly effective, but burdens backends, static analyzers, and verification frameworks, as now these also have to take into account low-level targets.

The framework presented in this paper intends to verify *functional requirements* (e.g. execution is bounded and error-free) and *non-functional requirements* (e.g. performance goals are reached) of network protocols in P4, in order to support new users of the language, and to provide insights and dynamic feedback to developers of the language ecosystem.

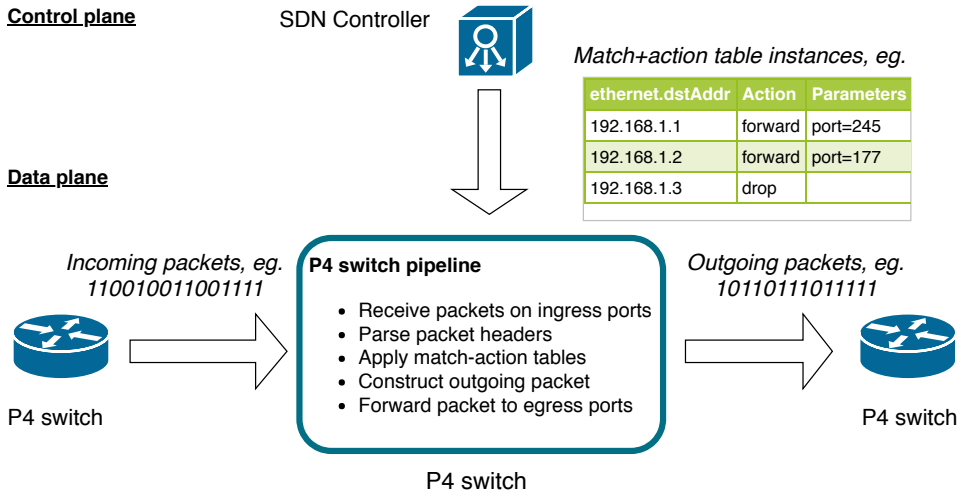


Figure 1: Application context of P4 pipelines

About P4

In Figure 1, the application context of pipelines defined in the P4 language is depicted. Being network protocols, P4 programs describe the packet filtering and forwarding activities to be performed by network devices. Unprocessed packets enter the pipeline on the *ingress-ports*, and processed packets exit on the *egress-ports*. In most pipelines, packets (bit-sequences) get *parsed* (i. e. headers and payload are identified) by a user-defined state machine, then packet headers are matched against one or more key-value stores (called *match+action tables*) as specified by the control flow definition. When a match is found, the corresponding *actions* are performed to modify the packet or the program state, or to trigger device-dependent side-effects (e.g. sending a message to the SDN controller). Match+action tables (successors of conventional routing tables) are received from the SDN controller at runtime and may change repeatedly during the operation of the switch.

Figure 2 depicts a small L2 routing protocol in P4. Here, packets get parsed, then they are matched based on their destination address and a suitable egress port is selected according to the match+action table instance. After the processing, the now modified packet gets forwarded to its destination.

```
#include <core.p4>
#include <v1model.p4>

header ethernet_t{
    bit<48> dstAddr;
    bit<48> srcAddr;
    bit<16> etherType;
}
ethernet_t ethernet;

parser ParserImpl(packet_in packet,
                  out headers hdr,
                  inout metadata meta,
                  inout standard_metadata_t smeta){
    state start {
        packet.extract(hdr.ethernet);
        transition accept;
    }
}

control Ingress(inout headers hdr,
                inout metadata meta,
                inout standard_metadata_t smeta){
    action forward(bit<9> port) {
        standard_metadata.egress_spec = port;
    }

    table dmac {
        actions = { forward; }
        key = { hdr.ethernet.dstAddr: exact; }
        size = 512;
    }
    apply { dmac.apply(); }
}

control DeparserImpl(packet_out packet,
                     in headers hdr){
    apply { packet.emit(hdr.ethernet); }
}

SwitchImpl(ParserImpl() Ingress(),DeparserImpl())
                                              main;
```

Figure 2: A simple L2 routing protocol in P4 that forwards packets based on their destination address according to the current content of the routing table.

Motivation

While software gives increased power for engineers to express their intent, the price of rising architectural complexity is the prevalence of faults and performance anomalies in shipped products. For example, the survey of Kim et. al [2] concludes that testing correctness, and guarantees of service quality (e.g. performance) are the most important problems in configuration automatisation. Efficient development requires improved tooling for developers in order to develop, troubleshoot and fine-tune efficient network setups proactively in development-time instead of deployment-time. On the other hand, automatised troubleshooting can also be utilised by the SDN controller automatically in deployment-time lessening the need for human intervention. Apart from providing tooling support for developers, performance estimations enable backend developers (e.g. T4P4S compiler project [3] at ELTE University, Hungary) to compare efficiency of the compilation output against a theoretical optimum. By getting a clear picture on the challenges of developing optimal backends for P4, developer experience can then be used in a feedback loop to improve and extend the language itself.

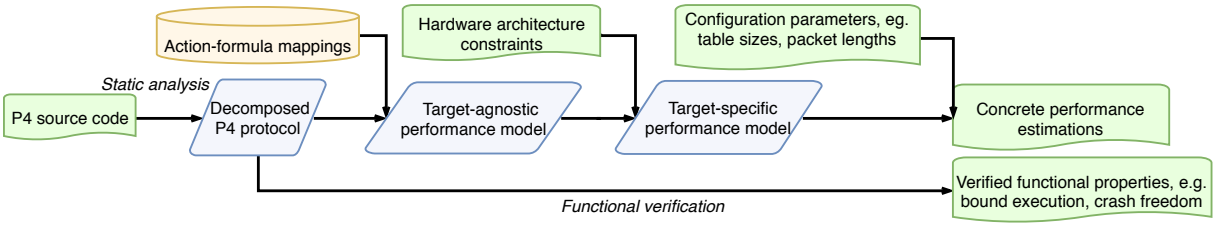


Figure 2: Dataflow diagram depicting inputs, outputs and intermediate results of the P4 verification framework in development

Related work

Our current work is only concerned with verification of switch behaviour in isolation. Before programmable switches, device-level verification meant model checking hardware implementations of protocols, and this was feasible as the protocols to verify were relatively simple. With arbitrarily complex protocols in the SDN forwarding plane appeared new research challenges too. Examples of interesting works on verification of software protocols are VigNat [4] implemented in C; verification of a data center protocol in P4 [5]; and verification of P4 programs by first compiling them to Datalog [6]. We are yet unaware of any approach that is able to automatically verify performance requirements on P4 protocols.

Methodology

The framework presented in this paper analyses P4 protocols accompanied by sufficient HW architecture descriptions to verify satisfaction of functional requirements (execution is bounded and crash-free) and non-functional requirements (performance goals are reached). In this paper, we intend to examine, synthesise, and build on two compositional methodologies in order to reach both goals.

In [7], the authors manually analyse the Ethernet protocol and a specific hardware architecture, which are then synthesised into a sequence of primitive packet processing actions called *elementary operations (EOs)* in order to quantify performance. We intend to automatise and apply the methodology in this work to P4 pipelines, as these can also be naturally decomposed into *primitive actions* and *extern objects*, which are dependent on the target network device.

To guarantee well-foundedness of the performance models, we need to show that the verified protocols always halt in an accepting program state. In [8], the authors show that *crash-freedom*, *bounded execution* and certain filtering properties can be proved by symbolically executing protocols of the Click network programming language. The authors utilise the composability of Click to avoid state-space explosion and to effectively linearise the cost of the search. They perform the search for problematic memory states in small pipeline components instead of the whole pipeline. As P4 pipelines are also compositional, the aforementioned idea applies to P4 similarly: to reduce the search space, we can search each parser blocks, control blocks, and user-defined actions in isolation, and only perform checks on the whole pipeline.

Figure 2 depicts a high-level view on the dataflow of the proposed method. The inputs of our method (to be supplied by its users) are the P4 protocol source code, the hardware architecture description, and specific network configuration parameters. The framework verifies correctness of the source and other functional requirements, followed by performance analysis. The performance estimates resulting from this analysis can then be used to verify that non-functional requirements are also satisfied. Building on the aforementioned research also allows us to validate our framework against the results in those works.

Conclusions

Scalability and vendor-independence gets more and more valuable for stakeholders in networking technologies, and at the same time vendors are required by the market to keep development costs low by moving from hardware to software. As a result, new technologies are emerging aiming to make the SDN data plane more programmable. Among them is P4, a language for describing switching pipelines. Our ongoing research effort aims to develop a framework to verify functional and non-functional requirements of network protocols defined in P4. In this paper, we presented a methodology based on [7] and [8] to check *crash-freedom* and *bounded execution* properties of P4 protocols, and estimate *lower and upper bounds* of packet processing time in order to verify that performance goals were reached. We believe that future iterations of this technology will bring user experience and efficiency of P4 users to new levels, and also provide deeper insights to developers of the language ecosystem.

Acknowledgements

The research has been supported by the European Union, co-financed by the European Social Fund (EFOP-3.6.2-16-2017-00013).

References

[1] D. Kreutz, F. M. V. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig. Software-Defined Networking: A Comprehensive Survey. *Proceedings of the IEEE*, 103(1):14–76, Jan 2015.

[2] Hyojoon Kim, Joshua Reich, Arpit Gupta, Muhammad Shahbaz, Nick Feamster, and Russ Clark. Kinetic: Verifiable Dynamic Network Control. In *Proceedings of the 12th USENIX Conference on Networked Systems Design and Implementation*, NSDI'15, pages 59–72, Berkeley, CA, USA, 2015. USENIX Association.

[3] Sándor Laki, Dániel Horpácsi, Péter Vörös, Róbert Kitlei, Dániel Leskó, and Máté Tejfel. High speed packet forwarding compiled from protocol independent data plane specifications. In *Proceedings of the 2016 ACM SIGCOMM Conference*, SIGCOMM '16, pages 629–630, New York, NY, USA, 2016. ACM.

[4] Arseniy Zaostrovnykh, Solal Pirelli, Luis Pedrosa, Katerina Argyraki, and George Candea. A Formally Verified NAT. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, SIGCOMM '17, pages 141–154, New York, NY, USA, 2017. ACM.

[5] Lucas Freire, Miguel Neves, Alberto Schaeffer-Filho, and Marinho Barcellos. Poster: Finding vulnerabilities in p4 programs with assertion-based verification, 10 2017.

[6] Nick McKeown, Dan Talayco, George Varghese, Nuno Lopes, Nikolaj Bjorner, and Andrey Rybalchenko. Automatically verifying reachability and well-formedness in p4 networks. Technical report, September 2016.

[7] A. Sapio, M. Baldi, and G. Pongrácz. Cross-Platform Estimation of Network Function Performance. In *2015 Fourth European Workshop on Software Defined Networks*, pages 73–78, Sept 2015.

[8] Mihai Dobrescu and Katerina Argyraki. Software dataplane verification. In *Proceedings of the 11th USENIX Conference on Networked Systems Design and Implementation*, NSDI'14, pages 101–114, Berkeley, CA, USA, 2014. USENIX Association.

Different Types of Search Algorithms for Rough Sets

Dávid Nagy, Tamás Mihálydeák, László Aszalós

Abstract:
Based on the available information in many cases it can happen that two objects cannot be distinguished. If a set of data is given and in this set two objects have the same attribute values, then these two objects are called indiscernible. This indiscernibility has an effect on the membership relation, because in some cases it makes our judgment uncertain about a given object. The uncertainty appears because if something about an object is needed to be stated, then all the objects that are indiscernible from the given object must be taken into consideration. The indiscernibility relation is an equivalence relation which represents background knowledge embedded in an information system. In a Pawlakian system this relation is used in set approximation. Correlation clustering is a clustering technique which generates a partition using search algorithms. In the authors' previous research the possible usage of the correlation clustering in rough set theory was investigated. In this paper the authors show how different types of search algorithms affect the set approximation.

Keywords: rough set theory, set approximation, data mining

Introduction

In many computer science applications objects are stored in a database. Each of these objects has a unique ID and other attributes. The ID of an object is only a technical tool and it does not represent any information about the object itself. In practice, two objects can only be distinguished if they differ in at least one known attribute value. If we want to decide whether an object belongs to an arbitrary set, based on the available data, then our decision affects the decision about all the objects that are indiscernible from the given object.

In this case if we would like to check whether an object is in an arbitrary set then the following three possibility appear:

- it is sure that the object is in the set if all the objects, that are indiscernible from the given object, are in the set;
- the object may be in the set if there some objects that are in the set and are indiscernible from the given object;
- it is sure that the object is not in the set if all the objects, that are indiscernible from the given object, are not in the set.

So the indiscernibility makes a set vague. The relation and the set theory based on it was developed by professor Pawlak.

From the theoretical point of view a Pawlakian approximation space (see in [1, 3, 2]) can be characterized by an ordered pair $\langle U, \mathcal{R}\rangle$ where U is a nonempty set of objects and $\mathcal{R}$ is an equivalence relation on U. In order to approximate an arbitrary subset S of U the following tools have to be introduced:

- *the set of base sets*: $\mathfrak{B} = \{B \mid B \subseteq U,$ and $x, y \in B$ if $x\mathcal{R}y\}$, the partition of U generated by the equivalence relation $\mathcal{R}$;
- *the set of definable sets*: $\mathfrak{D}_{\mathfrak{B}}$ is an extension of $\mathfrak{B}$, and it is given by the following inductive definition:
  1. $\mathfrak{B} \subseteq \mathfrak{D}_{\mathfrak{B}}$;
  2. $\emptyset \in \mathfrak{D}_{\mathfrak{B}}$;
  3. if $D_1, D_2 \in \mathfrak{D}_{\mathfrak{B}}$, then $D_1 \cup D_2 \in \mathfrak{D}_{\mathfrak{B}}$.
- *the functions* l, u form a Pawlakian approximation pair $\langle \mathsf{l}, \mathsf{u}\rangle$, i.e.
  1. $Dom(\mathsf{l}) = Dom(\mathsf{u}) = 2^U$
  2. $\mathsf{l}(S) = \bigcup\{B \mid B \in \mathfrak{B}$ and $B \subseteq S\}$;

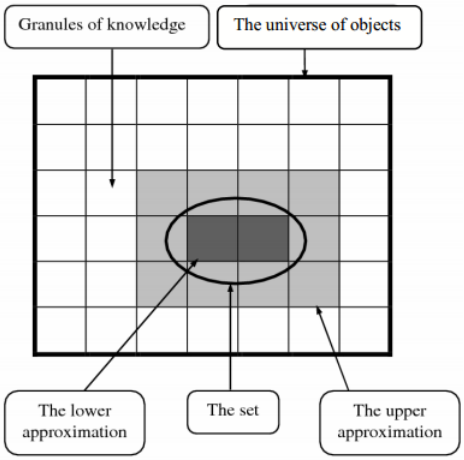


Figure 1: A rough set

3. $\mathrm{u}(S) = \bigcup\{B \mid B \in \mathfrak{B} \text{ and } B \cap S \neq \emptyset\}$.

U is the set of objects. $\mathfrak{B}$ is the system of base sets which represents the background knowledge. $\mathfrak{D}_{\mathfrak{B}}$ is the set of definable sets which defines how the base sets can be used in the set approximation. The functions l and u give the lower and upper approximation of a set. The lower approximation contains objects that surely belong to the set, and the upper approximation contains objects that possibly belong to the set.

In Fig 1 a set and its lower and upper approximation can be seen. Each rectangle denotes a base set.

Correlation clustering

Data mining is the process of discovering patterns and hidden information in large data sets. The goal of a data mining process is to extract information from a data set and transform it into an understandable structure for further use. Clustering is a data mining technique in which the goal is to group the objects so that the objects in the same group are more similar to each other than to those in other groups. In many cases the similarity is based on the attribute values of the objects. In most of them, some kind of distance is used to define the similarity. However, sometimes only nominal data are given. In this particular case distance can be meaningless. For example, what is the distance between a male and a female? In this case a similarity relation can be used, which is a tolerance relation. If this relation holds for two objects, we can say that they are similar. If this relation does not hold then they are dissimilar. It is easy to prove that this relation is reflexive and symmetric. The transitivity, however, does not hold necessarily. Correlation clustering is a clustering technique based on a tolerance relation (see in [5, 6, 7]).

Let V a set of objects and T the similarity relation. The task is to find an $R \subseteq V \times V$ equivalence relation which is *closest* to the tolerance relation.

A (partial) tolerance relation T (see in [10, 9]) can be represented by a matrix M. Let matrix $M = (m_{ij})$ be the matrix of the partial relation T of similarity:

$$m_{ij} = \begin{cases} 1 & i \text{ and } j \text{ are similar} \\ -1 & i \text{ and } j \text{ are different} \\ 0 & \text{otherwise} \end{cases}$$

A relation is called partial if there exist two elements (i, j) such that $m_{ij} = 0$. It means that if we have an arbitrary relation $R \subseteq V \times V$ we have two sets of pairs. Let R_{true} be the set of those pairs of elements for which the R holds and R_{false} be the one for which R does not hold. If R is partial, then $R_{true} \cup R_{false} \subseteq V \times V$. If R is total then $R_{true} \cup R_{false} = V \times V$.

A partition of a set S is a function $p : S \to \mathbb{N}$. Objects $x, y \in S$ are in the same cluster at partitioning p, if $p(x) = p(y)$. We treat the following two cases conflicts for any $x, y \in V$:

- $(x, y) \in T$ but $p(x) \neq p(y)$
- $(x, y) \notin T$ but $p(x) = p(y)$

The goal is to minimize the number of these conflicts. If their number is 0, the partition is called *perfect*. Given the T and R we call this value the distance of the two relations. The partition given this way, generates an equivalence relation. This relation can be considered as the closest to the tolerance relation.

The number of partitions can be given by the Bell number (see in [8]) which grows exponentially. For more than 15 objects, we cannot achieve the optimal partition by exhaustive search. In a practical case, a search algorithm can be used which can give a quasi optimal partition.

Similarity based rough sets

In practical applications, indiscernibility relation is too strong. Therefore, Pawlakian approximation spaces have been generalized using tolerance relations (symmetric and reflexive), which are similarity relations. Covering-based approximation spaces (see [11]) generalize Pawlakian approximation spaces in two points:

1. $\mathcal{R}$ is a tolerance relation;
2. $\mathfrak{B} = \{[x] \mid [x] \subseteq U, x \in U \text{ and } y \in [x] \text{ if } x\mathcal{R}y\}$, where $[x] = \{y \mid y \in U, x\mathcal{R}y\}$.

The definitions of definable sets and approximation pairs are the same as before. In these covering systems each object generates a base set.

Correlation clustering defines a partition. The clusters contain objects that are typically similar to each other. In our previous work (see in [4]) we showed that this partition can be understood as the system of base sets. The approximation space given this way has several good properties. The most important one is that it focuses on the similarity (the tolerance relation) itself and it is different from the covering type approximation space relying on the tolerance relation.

In reasonable time, correlation clustering can only be solved using search algorithms. However, each algorithm can provide different clusters. So the system of base sets can also be different. It is a natural question to ask, how the search algorithms can affect the structure of the base sets.

Acknowledgements

This work was supported by the construction EFOP-3.6.3-VEKOP-16-2017-00002. The project was co-financed by the Hungarian Government and the European Social Fund.

References

[1] Pawlak, Z.: Rough sets. International Journal of Parallel Programming 11(5), 341–356 (1982)

[2] Pawlak, Z., Skowron, A.: Rudiments of rough sets. Information sciences 177(1), 3–27 (2007)

[3] Pawlak, Z., et al.: Rough sets: Theoretical aspects of reasoning about data. Systern Theory, Knowledge Engineering and Problem Solving, Kluwer Academic Publishers, Dordrecht, 199l 9 (1991)

[4] Nagy, D., Mihálydeák, T., Aszalós, L.: Similarity Based Rough Sets, pp. 94–107. Springer International Publishing, Cham (2017), https://doi.org/10.1007/978–3–319–60840-2_7

[5] Bansal, N., Blum, A., Chawla, S.: Correlation clustering. Machine Learning 56(1-3), 89–113 (2004)

[6] Becker, H.: A survey of correlation clustering. Advanced Topics in Computational Learning Theory pp. 1–10 (2005)

[7] Zimek, A.: Correlation clustering. ACM SIGKDD Explorations Newsletter 11(1), 53–54 (2009)

[8] Aigner, M.: Enumeration via ballot numbers. Discrete Mathematics 308(12), 2544 – 2563 (2008), http://www.sciencedirect.com/science/article/pii/S0012365X07004542

[9] Mani, A.: Choice inclusive general rough semantics. Information Sciences 181(6), 1097–1115 (2011)

[10] Skowron, A., Stepaniuk, J.: Tolerance approximation spaces. Fundamenta Informaticae 27(2), 245–253 (1996)

[11] Yao, Y., Yao, B.: Covering based rough set approximations. Information Sciences 200, 91–107 (2012), http://www.sciencedirect.com/science/article/pii/S0020025512001934

Reconstruction of Rooted Directed Trees

Dénes Bartha

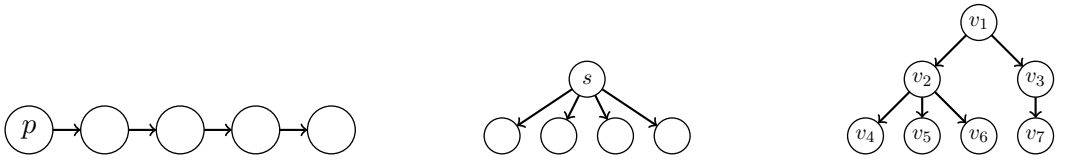
Abstract: Let T be a rooted directed tree on n vertices, rooted at v. The rooted subtree frequency vector (*RSTF-vector*) of T with root v, denoted by rstf(T, v) is a vector of length n whose entry at position k is the number of subtrees of T that contain v and have exactly k vertices. In this paper we present an algorithm for reconstructing rooted directed trees from their rooted subtree frequencies (up to isomorphism). We show that there are examples of nonisomorphic pairs of rooted directed trees that are *RSTF-equivalent*, s.t. they share the same rooted subtree frequency vectors. We have found all such pairs (groups) for small sizes by using exhaustive computer search. We show that infinitely many nonisomorphic *RSTF-equivalent* pairs of trees exist by constructing infinite families of examples.

Keywords: tree reconstruction, subtree size frequencies, rooted directed trees

Introduction

The main problem we investigate is the method of reconstructing an unlabeled rooted directed tree from the rooted subtree frequencies[3]. The motivation of the problem comes from mass spectrometry data analysis.

Definition 2. *Let $T = (V, E)$ be a rooted directed tree on $n \in \mathbb{N}^+$ vertices, rooted at vertex $v \in V$. The vector* rstf$(T, v) = [r_1, \ldots, r_n]$ *is called the rooted subtree frequency vector of T with root v, where each r_i shows the number of i-sized subtrees of T that contains v.*



rstf$(P_5, p) = [1, 1, 1, 1, 1]$ rstf$(S_4, s) = [1, 4, 6, 4, 1]$ rstf$(Q, v_1) = [1, 2, 5, 7, 7, 4, 1]$

$r(P_5, p) = 1 + x + x^2 + x^3 + x^4$ $r(S_4, s) = 1 + 4x + 6x^2 + 4x^3 + x^4$ $r(Q, v_1) = 1 + 2x + 5x^2 + 7x^3 + 7x^4 + 4x^5 + x^6$

Figure 1: P_4 denotes a path of length 4 rooted at p, S_4 a star with 4 leaves rooted at v_2, and a more complex tree Q on 7 nodes rooted at v_1. The appropriate *RSTF-vectors*, *RSTF-polynomials* are given below.

We can represent *RSTF-vectors* with polynomials by choosing the entries of the vectors as the appropriate coefficients of the polynomial.

Definition 3. *Let T be a rooted directed tree, v the root of T, with* rstf$(T, v) = [r_1, r_2, \ldots, r_n]$. *The* RSTF-polynomial *of T with root v, denoted by $r(T, v)$ is defined by $r(T, v) = r_1 + r_2x + r_3x^2 + \cdots + r_nx^{n-1}$.*

Figure 1 shows three examples on RFST-vectors and polynomials. The reason why we use *RSTF-polynomials* instead of vectors is that we can easily calculate the *RSTF-polynomial* from a given rooted directed tree graph structure.

Lemma 1. *Given a tree T with root v, one can calculate the* RSTF-polynomial *in $O(n^2)$ time using the following recursive formula:*

$$\text{rstf}(T, v) = \prod_{i=1}^{k} (1 + x \cdot \text{rstf}(T_i, v_i))$$

Note that here v has k children nodes where each T_i, v_i denotes the "child" subtree T_i rooted at v_i.

For example to calculate the *RSTF-polynomial* of the above given Q tree (given in Figure 1), we have to do the following calculation:

$$\begin{aligned}
\text{rstf}(Q, v_1) &= (1 + x \cdot \text{rstf}(Q_2, v_2)) \cdot (1 + x \cdot \text{rstf}(Q_3, v_3)) \\
\text{rstf}(Q_2, v_2) &= (1 + x \cdot \text{rstf}(Q_4, v_4)) \cdot (1 + x \cdot \text{rstf}(Q_5, v_5)) \cdot (1 + x \cdot \text{rstf}(Q_6, v_6)) \\
\text{rstf}(Q_4, v_4) &= \text{rstf}(Q_5, v_5) = \text{rstf}(Q_6, v_6) = 1 \\
\text{rstf}(Q_3, v_3) &= (1 + x \cdot \text{rstf}(Q, v_7)) \\
\text{rstf}(Q_7, v_7) &= 1
\end{aligned}$$

If we substitute back to $\text{rstf}(Q, v_1)$, and expand the product, the resulted polynomial's coefficient sequence gives the *RSTF-vector* of Q:

$$\text{rstf}(Q, v_1) = \left(1 + x\,(x+1)^3\right) \cdot \left(1 + x\,(x+1)\right) = \mathbf{1} + \mathbf{2}x + \mathbf{5}x^2 + \mathbf{7}x^3 + \mathbf{7}x^4 + \mathbf{4}x^5 + \mathbf{1}x^6$$

Note that we call the *RSTF-polynomial* polynomial *well formed* before expansion s.t. it consists of 1, $+$, x, $\cdot$ and $()$ symbols.

Reconstruction algorithm

Our main goal is to find an algorithm that could reconstruct from a given *RSTF-polynomail* the original rooted directed tree. Using Lemma 1 we can construct such an algorithm. The idea is that in each recursive step we factorize the given polynomial such that it produces a *well formed RSTF-polynomial*. We can also use this to determine whether the given polynomial is an actual *RSTF-polynomial*. One can make the following observation. A factorization of a *RSTF-polynomial* could be given, such that it has exactly a_1 factors (where a_1 denotes the x's coefficient). The first step is the polynomial factorization, the second one is finding the proper grouping:

$$\begin{aligned}
f &= a_0 + a_1 x + a_2 x^2 + \cdots + a_{n-1} x^{n-1} \\
&= p_1^{k_1} \cdot p_2^{k_2} \cdots p_m^{k_m} \\
&= (1 + x \cdot (f_1)) \cdot (1 + x \cdot (f_2)) \cdots (1 + x \cdot (f_{a_1}))
\end{aligned} \tag{16}$$

Lemma 2. *From a well formed RSTF-polynomial on degree $n-1$ we can reconstruct the corresponding tree structure by n steps.*

Figure **??** shows an example on how the algorithm works. In this short paper we omit the pseudocode and the analysis of the *RRDT algorithm*. An implementation of the RRDT algorithm written in sage, python can be found at https://github.com/denesbartha/RDRT

Isomorphism and Reconstructibility reults

The *RRDT algorithm* is able to reconstruct rooted directed trees up to isomorphism. Not surprisingly there are cases when a given input polynomial represents multiple rooted directed trees. Figure 3 shows two nonisomorphic rooted directed trees that share the same *RSTF-polynomial*.

Definition 4. *We call T and T' nonisomorphic rooted directed trees* RSTF-equivalent *iff they share the same* RSTF-polynomial.

The following equation shows the *well formed RSTF-polynomials* of R_7 and R_7' trees.

$$\begin{aligned}
&\text{rstf}(R_7, v_1) = \text{rstf}(R_7', v_2) \\
&= x^6 + 3x^5 + 4x^4 + 4x^3 + 3x^2 + 2x + 1 \\
&= (x^3 + x^2 + 1) \cdot (x^2 + x + 1) \cdot (x + 1) \\
&= \Big(1 + x \cdot \Big(1 + x \cdot \Big((1 + x) \cdot (1 + x)\Big)\Big)\Big) \cdot \Big(1 + x \cdot (1 + x)\Big) \\
&= \Big(1 + x \cdot \Big(\Big(1 + x \cdot (1 + x \cdot (1 + x))\Big) \cdot \Big(1 + x\Big)\Big)\Big) \cdot (1 + x)
\end{aligned}$$

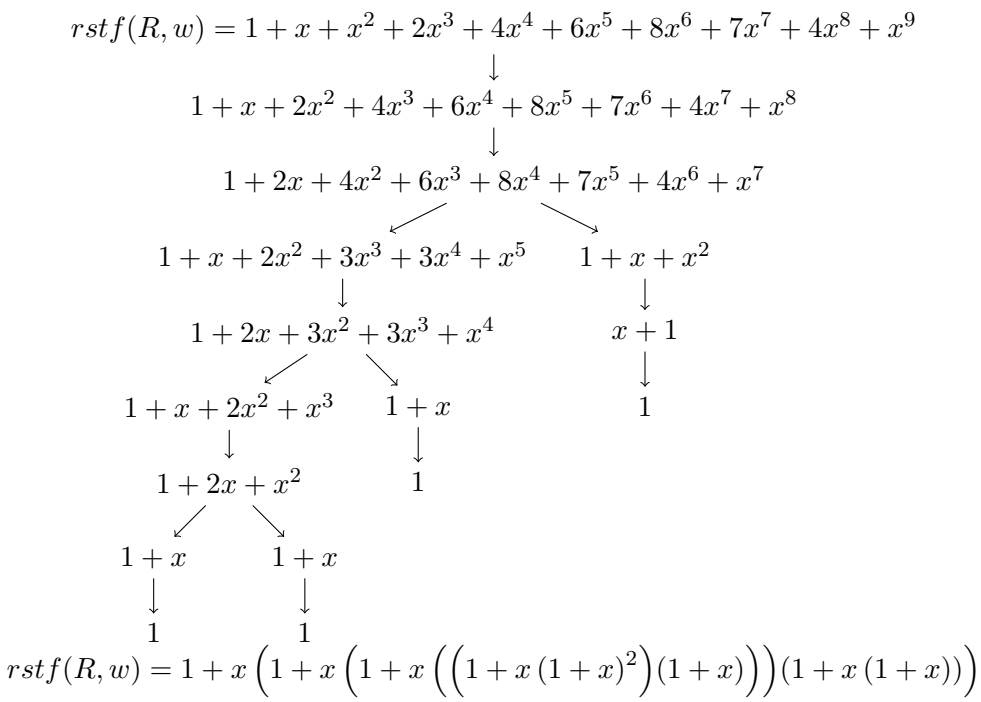


Figure 2: Given a polynomial $1 + x + x^2 + 2x^3 + 4x^4 + 6x^5 + 8x^6 + 7x^7 + 4x^8 + x^9$. Each node in the graph (except the root w) is constructed by and connected with the appropriate parent node.

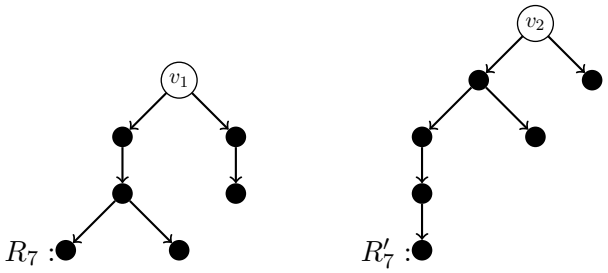


Figure 3: R_7 and R'_7 are two nonisomorphic rooted directed *RSTF-equivalent* trees

With exhaustive computer search we have determined all the nonisomorphic equivalence classes of rooted directed trees up to $n = 20$. Table 1 summarizes the results. We have also found infinite families of nonisomorphic rooted directed *RSTF-equivalent* trees (we omit these examples and proofs from this short paper).

Note that the maximum number of equivalence classes $ec(n)$ cannot exceed the number of nonisomorphic rooted directed trees $a(n)$, hence $\frac{ec(n)}{a(n)} \leq 1$. Also because there are infinite nonisomorphic rooted directed *RSTF-equivalent* tree classes exist $0 < \frac{ec(n)}{a(n)} < 1$, for $n \geq 7$. Our conjecture is that $lim_{n\to\infty} \frac{ec(n)}{a(n)} = 0$.

The project was supported by the European Union, co-financed by the European Social Fund (EFOP-3.6.3-VEKOP-16-2017-00001).

References

[1] B. Manvel, *On reconstructing graphs from their sets of subgraphs*, J. Combin. Theory Ser. B **21** (1976), 156–165.

[2] B. Manvel, *Reconstruction of trees*, Canad. J. Math. **22** (1970), 55–60.

$\mathbf{n}$	$\mathbf{a(n)}$	#equivalence classes	$\frac{\mathbf{ec(n)}}{\mathbf{a(n)}}$	Maximal size equivalence class	Entropy
3	2	2	1.0	1	0
4	4	4	1.0	1	0
5	9	9	1.0	1	0
6	20	20	1.0	1	0
7	48	47	0.97917	2	0.14855
8	115	112	0.97391	2	0.178
9	286	274	0.95804	2	0.25943
10	719	679	0.94437	2	0.3231
11	1842	1717	0.93214	3	0.3833
12	4766	4393	0.92174	4	0.42953
13	12486	11374	0.91094	4	0.47557
14	32973	29725	0.9015	5	0.51466
15	87811	78428	0.89315	7	0.54811
16	235381	208431	0.8855	8	0.57819
17	634847	557555	0.87825	11	0.60622
18	1721159	1499739	0.87135	11	0.63245
19	4688676	4054714	0.86479	15	0.65711
20	12826228	11011259	0.8585	16	0.68046

Table 1: second column: $a(n)$ - the number of unlabeled rooted trees with n nodes (OEIS - A000081); third column: the number of nonisomorphic equivalence classes; fourth column: the ratio of #equivalence classes to $a(n)$; fifth column: the maximal size equivalence class; sixth column: Shannon entropy of the equivalence classes

[3] D. Bartha and P. Burcsi, *Reconstructibility of trees from subtree size frequencies* Stud. Univ. Babes-Bolyai Math. 59 (2014), No. 4, 435-442

[4] D. Bartha, P. Burcsi, and Zsuzsanna Lipták, *Reconstruction of Trees from Jumbled and Weighted Subtrees* LIPIcs.CPM.2016.10

[5] I. Krasikov, Y. Roddity, *On a reconstruction problem for sequences*, J. Combin. Theory Ser. A **77** 2 (1997), 344–348.

[6] J. Acharya, H. Das, O. Milenkovic, A. Orlitsky, S. Pan, *String reconstruction from substring compositions*, arXiv:1403.2439v1

[7] J. Dudik., L. J. Schulman, *Reconstruction from subsequences*, J. Combin. Theory Ser. A **103** 2 (2003), 337–348.

Cognitive Enterprise and Cognitive Information Systems

Dóra Mattyasovszky-Philipp, Bálint Molnár

Abstract: The rapid change of world causes new challenges in the business environment and in many other operational areas of enterprises. Those challenges likely increase the chance for client or customer satisfaction meanwhile improve companies' efficiency, improve cost saving, optimize processes in operation and/or manufacturing, enhance security, reduce errors etc. Due to the complexity and to the rapid alterations within the environment, changes in the processes of enterprises are not able to answer these challenges easily in time, e.g. predicting the potential future events as one of the key elements of the long-term success, these events might have a direct impact on the organization future. Those required skills to manage the actual situation are rarely available in one single person or in one team, sometimes it is neither affordable or nor efficient, therefore one of the possible solution is to leverage the capability of a cognitive information system. The aim of the publication is highlight advantages of the cognitive information system. Help to understand and imagine the future of cognitive enterprise, cognitive operation, analyses in different segments, industries. In the areas presented, point to the potential benefits of the application and highlighting the importance of the support to achieve the business goals, realizing the corporate strategy. Cognitive information systems enable the corporate to leverage new technologies, supports effectiveness improvement. Demonstrate with example the possibilities what business process management provides, supported by the cognitive information system how to extend the boundaries of the operation of the enterprise. Challenges out of the environment require continuous improvement; therefore, a business process improvement that needs to adapt to the situation. Cognitive information system enables to reveal the gaps in the current business process, identifies the opportunities for development and supports the daily operation with business intelligence under the framework of business process management, CRM, and Kaizen methodology.

Keywords: Business Information Systems, Cognitive Information Systems, Cognitive Enterprise, Information Systems Modelling, Business Process Alignment.

Introduction

The reconstruction of binary matrices from their projections is a basic problem in discrete tomography. The concept of Cognitive Information Systems appeared as an intersection and shared area of Information Systems and Cognitive Science. Modeling, analyzing, and designing information systems in the most recent technological advancement makes it possible and - at the same time - requests for architectural and design principles that combine the development of technology and theories of Cognitive Science. We think of architecture of information systems within enterprises in the sense of Zachman [9] and TOGAF framework [4]. The technology advancement in the field of information systems can be captured in the concept of Data Science and semantics. Cognitive Science offers the analysis of cognitive tasks, i.e. the investigation of the decision making, reasoning skills of the examined subject whilst the subjects treat the set of complex information [7]. Interdisciplinary research on human expertise and domain specific cognition has yielded theoretical and methodological background that can be employed to create architecture and design principles that can assist in the realization of cognitive information systems [5]. Cognitive science studies "the principles, architectures, organization, and operation of natural and artificial intelligence systems" [2], thereby cognitive science is an intersection of scientific domains among artificial intelligence, psychology, linguistics, anthropology, neuron-sciences, and education. Enterprise operation, through business process management leveraging cognitive information system capability, that supports the organization to respond to the environment challenges. The research methodology and assessment of the available publications pursued the comparative study pattern, i.e. studying and comparing the result published in articles, then a case study paradigm has been used for observing and monitoring the work with a cognitive information system during daily operation.

LITERATURE SURVEY

Cognitive information systems can be regarded as an interdisciplinary research topic that has emerged as an interaction between information systems and cognitive sciences including psychology, neuronscience, cognitive modeling, cognitive ergonomics, linguistics, biology, anthropology, and various branches of artificial intelligence [5]. Taking into account of the cognitive capability: it is an abstract notion that incorporates the temporal and contextual environment to be considered within the analysis and design of cognitive information systems. Cognitive information system are socio-technological systems. What makes a system cognitive is the following according to Hurwitz: "Three important concepts help make a system cognitive: contextual insight from the model, hypothesis generation (a proposed explanation of a phenomenon), and continuous learning from data across time." [3]. Wang postulates that a "denotation mathematical" approach is required that own structures, tools, and methods beyond the traditional mathematical logic [8]. Ogiela [6] collected areas of application of informatics and information technology where cognitive information processing techniques are incorporated into various systems as business information systems, biomedical systems, identity and access management, etc. that undoubtedly has impacts on business and delivers advantages, however none of them are covering enterprise environment in general. Currently, the human, carbon agent extended with a silicon agent with embedded knowledge-bases and reasoning capabilities. The two facets of data interpretation should be fitted together, at least in the sense of rough sets; if they cannot be reconciled then the trial to integrate the different views of data to understand them is not successful.

Illustrative example of cognitive business operation and analyses

The new operational ways with new business processes, business process management and process re-engineering, agile corporate management, process documentation, are set of activities in order to manage your processes in a fast, error-free, and cost-effective way. Strengths of cognitive computing, the ability to generate value from moving data, like stock exchange data that change in a folio-like manner where processing speeds are key. Real-time analysis involves the part that, if not realized, the value of the data is lost or significantly reduced. The use of the cognitive information system in various field always affects enterprises, and through the networks of business and society has effect on other layers of society. At the University Medical Center, Groningen, Netherlands the paper based administration system showed numerous errors over a 5-months period. These errors were transcription mistakes causing preventable drug event, temporary harms, prolonged hospital admissions, including cases that were life-threatening and fatality [11]. Based on the results of developing systems that reflect the physical, cognitive and social needs and goals of a person or team in the context of the technology, environment and culture with which they operate, positively impacted all of the negatives created by the paper based system as medical errors, adverse events, reduction of mortality, and complications [12].The cognitive information systems like Watson completely different from the traditional information system. The result of the analyses completed with Watson might initiate the transformation of the enterprise, changes on the governance, redefines the mode of operation, and impacts changes in the roles and responsibility in the organization. The customer experience is becoming more important to businesses as a differentiator; the sophisticated grow and increased demand did not pair with loyalty [10]. During of CRM system implementation, cognitive information system that analyses business processes, customer behavior, could help to the management or to the employees in buy in, utilizing visualization capability, demonstrating impacts and factors in a combined view. As is analyses helps to understand the actual situation, and calls attention for extremism. Cognitive information system ability to analyze deviations, highlights weaknesses and possible further development with actions, provides information on impacts drivers and correlation within inputs and outputs and other factors. Those outputs get the attention paid by the management and calls for actions. Kaizen methodology combined with the cognitive information system provides business intelligence, a cognitive business process management, which ensures several opportunities for the corporate for continuous improvement. Actions derivate form the combined methodology might pull immediate advantages. As Kaizen methodology as cognitive computing supports the enterprise on various level form employee to management, providing differentiate view with various visualization and level of information about the enterprise itself and its environment.

Analyses with Cognitive information system

Cognitive information systems with different cognitive level, provide different level of added-value. Descriptive Analytics: Historical and current data elaboration and analytics, which is lack of the above mentioned value add, therefore any traditional/technical analyses and traditional information system basic capability, sometimes with human collaboration. Predictive Analytics: Historical, current and real-time data elaboration and analytics, based on statistical predictive modeling, where the process includes data mining based on semantical analyses and machine learning. The output is presenting the possible future as value add during the decision making that includes real-time data. Prescriptive Analytics: The outcome defines go and no go rules regarding to the future activity highlighting the possible outcomes for it based on historical, current and real-time data evaluation. Advantage to simplify the decision process. Risk might carried by the rigidity of prescriptions. Cognitive Computing with machine learning Carbon agent (human) and silicon agent (CIS) collaboration for problem solving. Historical and real time Big data analytics from multiple sources, with prediction. Risk might carry by the carbon agent collaboration during learning session.

FUTURE RESEARCH DIRECTION

Considering the Zachman framework as an overarching approach, a cognitive information system belongs to the views of Business Analyst, Strategic Planner, System Analyst and System Designer. The cognitive information system is involved in the process/function and data perspectives, exploit the achievements of data science stepping beyond the traditional stimulus-response metaphor for conceptualizing of functional services. It is an adaptive system that can handle the dynamically changing business environment and can incorporate the alteration of business processes by a way that the realization of changes is implemented in a systematic and consistent manner through the three seasons of business processes, i.e. analysis, design and implementation/operation [1]. The above presented analysis about the concept of cognitive information systems in enterprises pinpoints the recent advancements. Enterprises come across regularly business problems, the data analytics as applied Data Science is a new source of information for enterprises to chase efficiency and effectiveness in their operations. The objective of data analytics procedure is to acquire knowledge out of collected data through discovering associations and relationships.

CONCLUSION

According to the most recent literature the definition of cognitive information systems refers to the cognitive architectures, data processing, models for insights hypotheses for new phenomenon and opportunity for the silicon agents for the continuous learning. The scientific and technological literature contains the description of massive cognitive information systems with the enormous resource requirements that may be accessible by huge international enterprises as tenants of cloud services. The subject of the research is two-pronged, on the side of the practice, the main aim is to grasp the notion of cognitive information systems that can be used for micro, small and medium enterprises, on the other side of formal approach the goal is to establish a model that define the business and information architecture, the major services exploiting the formal and semi-formal tool set of computer science.

Acknowledgements

The project has been supported by the European Union, co-financed by the European Social Fund (EFOP-3.6.3-VEKOP-16-2017-00002).

References

[1] Bell, M.: Service-oriented modeling (SOA): Service analysis, design, and architecture. John Wiley Sons, (2008).

[2] Grigoriev, E.A.: The cognitive role of intuitive hypotheses and visual image of simulated reality. CASC' 2001, pp: 5-16, (2001).

[3] Hurwitz J., Kaufman, M., Bowles, A.: Cognitive Computing and Big Data Analytics. John Wiley Sons, Inc. 10475 Crosspoint Boulevard Indianapolis, IN 46256, (2015).

[4] Josey A.: TOGAF Version 9.1 A Pocket Guide. Van Haren, (2011).

[5] Miller G.A.: The cognitive revolution: a historical perspective. Trends in cognitive sciences, 7(3), 141-144, (2003).

[6] Ogiela, L., Ogiela, M.R.: Advances in cognitive information systems (Vol. 17). Springer Science Business Media, 17-18, (2012).

[7] Rogers, Y., Sharp, H., Preece, J.: Interaction design: beyond human-computer interaction. John Wiley Sons, (2011).

[8] Wang, Y.: The theoretical framework of cognitive informatics. International Journal of Cognitive Informatics and Natural Intelligence (IJCINI), 1(1), 1-27, (2007).

[9] Zachmann J,.A: A framework for information systems architecture. IBM Systems Journal, 26(3), 276-292, (1987).

[10] Kostojohn S.,et.al.:CRM Fundamentals Apress, (2011.)

[11] Van Doormaall, J. E., et al. Medication errors: the impact of prescribing and transcribing errors on preventable harm in hospitalised patients. BMJ Quality and Safety 18.1 (2009): 22-27.

[12] Lawler E.K., et al., 2011. Cognitive ergonomics, socio-technical systems, and the impact of healthcare information technologies. International Journal of Industrial Ergonomics, 41(4), pp.336-344.

Who Are You not gonna Call?
A Definitive Comparison of Java Static Call Graph Creator Tools

Edit Pengő, Zoltán Ságodi, Ervin Kóbor

Abstract: Call graphs are fundamental for advanced, interprocedural control flow and data flow analysis tasks. In dynamic languages like Java, which allows polymorphism and reflection constructing a static call graph can be complicated. A missing or imprecisely connected edge might misguide the following algorithms causing errors in the overall analysis.

In this paper, we have collected six static analyzer tools for Java and performed a qualitative comparison on the call graph they generate. As part of the comparison, we introduced a method for pairing different notations of the same functions. We evaluated the collected tools on three open-source Java projects and on a small example containing most of the relevant Java language features. The results revealed several language structures that were handled differently by the static analyzers, which led to a difference in the created call graphs as well.

Keywords: Java, Call Graph, Static Code Analysis

Introduction

Developing quality software is a complex task and the unclear requirements, the underestimated efforts, and the strict deadlines make it even more difficult. Therefore, every software contains bugs even after its release. Static code analyzer tools help programmers produce more maintainable code and eliminate flaws early on by automatically analyzing the source code and identifying the potentially problematic places. However, the capabilities of these tools can differ considerably depending on the complexity of the representations and algorithms they use. Call graphs are building blocks for, for example, static control flow and data flow analysis. Therefore, it is crucial for the subsequent analyses to have a carefully constructed call graph especially if we consider dynamic language constructions as function pointers and polymorphism. In this work, we compared six open source static Java analyzer tools based on the call graphs we generated with them. We used three real life Java projects as test inputs along with a sample program containing the very essence of Java language. We introduced a comparing mechanism for the various graph representations provided by the tools and studied the results to identify Java language constructions that usually cause differences among the call graphs.

Murphy et al. [6] carried out a similar study of five static call graph creator for C in 1996. They identified significant differences in how the tools handled typical C constructs like macros. Rountev [9] built a framework to analyze differences in static and dynamic call chains in Java. They constructed static call chains from the edges of static call graphs. Reif et al. [8] studied the usability of call graph creation algorithms for Java libraries. They showed that there can be significant differences in the graphs depending on which algorithm was used. Lhoták proposed the importance of comparability among static analyzer tools. In his 2007 study [4] he built a general framework to compare static and dynamic call graphs, discussed the challenges of comparison and presented an algorithm to find the causes of differences in call graphs. There are several studies about dynamic call graph based fault detection like the work of Eichinger et al. [3] who created and mined weighted call graphs to achieve more precise bug localization. Liu et al. [5] constructed behavior graphs from dynamic call graphs to find noncrashing bugs and suspicious code parts with a classification technique.

The rest of the paper is organized as follows. Section gives a general description about call graphs and the tools we compared. The results of the comparison is presented in Section . Finally, we conclude the study in Section .

Call graph

Call graphs represent control flow relationships among the functions of a program. The nodes of the graph are the functions that are connected with directed edges. An edge from node a to node b indicates that function a invokes function b. Two types of call graphs can be distinguised: *static* and *dynamic* call graphs. Static call graphs are composed during the static analysis of the code. Ideally, they are conservative in a way that they contain every possible function call that can be realized during

the execution of the program. Considering dynamic language structures like function pointers and polymorphism, it is clear that constructing static call graphs is not a trivial task. Code analyzers can implement several algorithms [2, 10, 1] that address the difficulties of dynamic linking and make the static call graph more precise. Dynamic call graphs are constructed from call traces recorded during a concrete execution of the program, therefore they are the subsets of the ideal static call graphs. Although they contain proper dynamic binding information, they show only those functions and calls that were used during execution.

Tools

First, we examined many Java static analyzer tools that could generate or could be modified to generate call graphs. Although many seemed promising, a lot of them had to be eliminated for various reasons. We searched for free, widely available, open-source programs that were robust enough to analyze complex, real-life Java systems. Countless plug-in based call graph visualizers are available in IDEs like Eclipse, however as their output is mostly visual they were not usable for our purposes. Finally, we have selected six tools for our comparison.

- *OpenStaticAnalyser*[1] (OSA) is an open-source multi-language deep static analyzer framework developed by Department of Software Engineering, University of Szeged. It calculates source code metrics, detects code clones and finds coding rule violations in the source code. We extracted the static call graph by extending OSA with a visitor.
- *Soot*[2] is a language manipulation and optimization framework developed by the Sable Research Group at the McGill University. Although its latest official release was in 2012, the project is active with regular commits and nightly builds.
- *Spoon*[3] is an open-source library for the analysis and transformation of Java source code [7]. The project is well documented and actively developed with Java 9 support as well. We expanded it with a visitor for the Java metamodel it provides to generate call graphs.
- *WALA* [4] is a static analyzer for Java bytecode and JavaScript. It was originally developed at the IBM T.J. Watson Research Center and it is still maintained as an open source project. Wala has a built-in call graph generator functionality that we feed with all the methods of the program as entry points.
- *Java Call Hierarchy Printer*[5] (CHP) is a spoon based method hierarchy printer. It is a small GitHub project with only one contributor which has seemingly been inactive since 2015. To gain call graphs, we have created a wrapper tool that feeds CHP with the method names of the analyzed code and processes the printed call hierarchies.
- *Gousiosg call graph builder* [6] is an Apache BCEL based Java Call Graph Utility for generating static and dynamic call graphs. The project has two contributors and the last commit was made in 2017. It works on jar files and produces the output in a simple text format that we processed with a wrapper program.

Results

The examined six tools gave different outputs where not only the file format differed but also the names of the Java methods did as well. In order to be able to compare the results, the different notations of the methods names had to be unified. It was an easy task for "basic" methods, but it was very difficult for coding-features like *inner*, *anonymous*, *generic classes* and *lambdas*. For example, an anonymous class inside an anonymous class had different names (`Example$1$1` and `Example$2`). Therefore we developed a common method representation and transformed every method name to it. To validate the transformation we created a sample Java program that contained most of the various class and method constructs Java 8 allows and manually verified the results. We executed the six tools on the sample project and first compared how similar the found methods are. Table 1 shows the results where the diagonal cells present the number of methods found by the given tool and every other cell in a row shows how many percent of its methods was found by the other tool. The tools found almost the same number of methods and only the constructors or initializations differed except CHP who could not properly

[1] OpenStaticAnalyser GitHub Page: https://github.com/sed-inf-u-szeged/OpenStaticAnalyzer
[2] Sable/Soot GitHub Page: https://github.com/Sable/soot
[3] Spoon HomePage: http://spoon.gforge.inria.fr
[4] Wala HomePage: http://wala.sourceforge.net/wiki/index.php/Main_Page
[5] Call Hierarchy Printer GitHub Page: https://github.com/pbadenski/call-hierarchy-printer
[6] gousiosg/java-callgraph GitHub Page: https://github.com/gousiosg/java-callgraph

handle lambdas and generic classes either. Our method unification gave good results because, apart from CHP, at least 75% of the methods can be paired.

	Soot	OSA	Spoon	CHP	Gous.	WALA
Soot	**64**	85.9%	85.9%	70.3%	89.0%	90.6%
OSA	77.5%	**71**	95.8%	71.8%	85.9%	87.3%
Spoon	77.5%	95.8%	**71**	71.8%	85.9%	87.3%
CHP	75.0%	85.0%	85.0%	**60**	83.3%	80.0%
gous.	83.8%	89.7%	89.7%	73.5%	**68**	91.2%
WALA	79.5%	84.9%	84.9%	65.8%	84.9%	**73**

Table 1: Common methods in the sample proj.

	Soot	OSA	Spoon	CHP	Gous.	WALA
Soot	**267**	44.2%	44.2%	28.5%	47.6%	55.8%
OSA	80.3%	**147**	98.6%	57.1%	87.1%	87.1%
Spoon	79.2%	97.3%	**149**	56.4%	85.9%	85.9%
CHP	56.7%	60.4%	60.4%	**139**	59.7%	57.6%
gous.	79.9%	80.5%	80.5%	52.2%	**159**	84.3%
WALA	86.6%	74.4%	74.4%	46.5%	77.9%	**172**

Table 2: Common calls in the sample proj.

Comparison of Call Edges

After unifying method names, we were able to compare calls. Table 2 presents the comparison results of the calls on the sample project where a diagonal contains the number of calls the given tool found and the other cells in its row show how many percent of its calls were found by the other tools. As we can see, all tools but Soot identified more or less the same number of calls because Soot connects every instantiation to `Object` initialization, therefore every `new` expression yields a calls. Therefore, its low comparison values are expected as well because their maximum values (when all methods found by the other tools could have been paired) are around 60%. CHP also had worse results because it could not connect methods of anonymous classes to the right caller. On the other hand, the other OSA, Spoon and Gousiosg found very similar calls and the different handling of static and dynamic initializations and constructor calls caused the most difference.

Table 3 presents the results of the three examined projects: Joda 2.9.9 (85,911 KLOC), Apache Commons Math 3.6.1 (208,876 KLOC) and the Java ASG module of OpenStaticAnalyzer (44,453 KLOC). The results of CHP is missing because it did not work on larger systems. OSA, Spoon and Gousiosg found similar number of calls while Soot and WALA idetified much more calls. Besides, the three tools handle the "special calls" (e.g. the initializations and system classes) quite a similar way, therefore their calls also coincide. Since Gousiosg identifies more calls its results seem worse but among its calls can be found most of the calls OSA and Spoon found (see gous. column). The calls of Soot and WALA not only differ from the previous ones, but their calls do not correspond with each other. Its main reasons are that Soot includes every `Object` initialization and possible inherited classes while WALA connects almost every method from standard packages which are filtered out by the other tools. On the other hand, in case of OpenStaticAnalyzer the result of Soot is much closer to the other tools because in this system there were only several "special" (e.g. lambda) classes.

Conclusion

We collected six open-source Java analyzers that were able to or could be extended to generate call graph. To be able to compare the tools, we developed a program that unified the different method names the tools used and compared the call graph as well. We evaluated the six tools on a sample project and on three open-source systems and found that CHP did not work on large systems while only three out of the five tools gave similar results.

In the future, we plan to perform a more detailed analysis to better understand the differences of the call graphs. For this, we will utilize a graph database to efficiently use general graph similarity algorithms for a deeper comparison.

Acknowledgements

This research was supported by the project "Integrated program for training new generation of scientists in the fields of computer science", no EFOP-3.6.3-VEKOP-16-2017-0002. The project has been supported by the European Union and co-funded by the European Social Fund.

	Soot	OSA	Spoon	Gous.	WALA	Soot	OSA	Spoon	Gous.	WALA	Soot	OSA	Spoon	Gous.	WALA
Soot	**27,086**	12.7%	13.2%	13.6%	53.0%	**42,214**	8.3%	8.5%	9.4%	33.2%	**69,481**	13.3%	13.4%	13.2%	49.7%
OSA	34.3%	**9,995**	99.9%	86.9%	75.0%	17.5%	**20,059**	94.8%	84.9%	78.0%	67.5%	**13,643**	100.0%	91.2%	85.9%
Spoon	35.1%	98.0%	**10,189**	87.0%	75.4%	17.5%	92.7%	**20,494**	86.9%	74.2%	67.7%	99.0%	**13,775**	91.2%	85.1%
gous.	27.4%	64.5%	65.7%	**13,482**	53.6%	11.2%	48.0%	50.2%	**35,476**	45.3%	35.6%	48.5%	48.9%	**25,666**	44.5%
WALA	36.2%	18.9%	19.4%	18.2%	**39,657**	18.3%	20.5%	19.9%	21.0%	**76,496**	67.7%	23.0%	23.0%	22.4%	**50,990**

Table 3: Calls and common calls of Joda, Apache Commons Math and OpenStaticAnalyzer

References

[1] David F. Bacon and Peter F. Sweeney. Fast Static Analysis of C++ Virtual Function Calls. *SIGPLAN Not.*, 31(10):324–341, October 1996.

[2] Jeffrey Dean, David Grove, and Craig Chambers. Optimization of Object-Oriented Programs Using Static Class Hierarchy Analysis. In *ECOOP'95 – Object-Oriented Programming, 9th European Conference, Åarhus, Denmark*, pages 77–101. Springer Berlin Heidelberg, 1995.

[3] Frank Eichinger, Klemens Böhm, and Matthias Huber. Mining Edge-Weighted Call Graphs to Localise Software Bugs. In *Machine Learning and Knowledge Discovery in Databases*, pages 333–348, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[4] Ondřej Lhoták. Comparing call graphs. In *Proceedings of the 7th ACM SIGPLAN-SIGSOFT Workshop on Program Analysis for Software Tools and Engineering*, pages 37–42, 2007.

[5] Chao Liu, Xifeng Yan, Hwanjo Yu, Jiawei Han, and Philip S. Yu. Mining Behavior Graphs for "Backtrace" of Noncrashing Bugs. In *SDM*, 2005.

[6] Gail C. Murphy, David Notkin, William G. Griswold, and Erica S. Lan. An Empirical Study of Static Call Graph Extractors. *ACM Trans. Softw. Eng. Methodol.*, 7(2):158–191, April 1998.

[7] Renaud Pawlak, Martin Monperrus, Nicolas Petitprez, Carlos Noguera, and Lionel Seinturier. Spoon: A Library for Implementing Analyses and Transformations of Java Source Code. *Software: Practice and Experience*, 46:1155–1179, 2015.

[8] Michael Reif, Michael Eichberg, Ben Hermann, Johannes Lerch, and Mira Mezini. Call Graph Construction for Java Libraries. In *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, FSE 2016, pages 474–486, New York, NY, USA, 2016. ACM.

[9] Atanas Rountev, Scott Kagan, and Michael Gibas. Static and Dynamic Analysis of Call Chains in Java. *SIGSOFT Softw. Eng. Notes*, 29(4):1–11, July 2004.

[10] Vijay Sundaresan, Laurie Hendren, Chrislain Razafimahefa, Raja Vallée-Rai, Patrick Lam, Etienne Gagnon, and Charles Godin. Practical Virtual Method Call Resolution for Java. *SIGPLAN Not.*, 35(10):264–280, October 2000.

Agile method in education

Enikő Ilyés

Abstract: Nowadays agile software development has become popular. This fact allows software development programs in higher education institutes to refresh their curriculum so that students can be prepared for the use of agile methods. One approach to this is to apply Scrum in software development courses. But this constitutes a considerable challenge too. Since students only work together 1,5 hours a week, they don't have the possibility to practice Scrum daily. They don't have enough time for detailed sprint planning, assigning team members individually since that's the requirement of the educational system is nonagile, etc. Based on international research and case studies from Eötvös Loránd University, this article presents some critical questions and good practices regarding using Scrum in a software development course.

Introduction

Using Scrum in a traditional software development course (working 1,5 hours together and 6,5 hours alone, at home per week) raises many questions.

An important argument against using traditional Scrum is that students meet only once per week, so they don't have the possibility to practice Scrum every day. What length should a sprint have in this situation? This decision influences the time needed for sprint planning, the frequency of demos and reviews. These are important from the education point of view because of the lack of experience the students have in software development and management. The special roles of Scrum must be carefully chosen too. How can the appropriate student be chosen for a role if the teachers don't know the students ahead of time? Should the teachers fulfill all special roles? How can the students be motivated to function as a self-organized, independent team, having common responsibilities but still be evaluated individually since that is the requirement of the educational system? The boundaries of a software development course control the size, complexity of the software to be realized during the semester hence choosing the right project is very important. Finally, the biggest challenge is to transmit the agile attitude that runs along Scrum roles, events, products, especially because of the time constraints that an education course has

International research

Teachers across the globe highlight a few aspects to be considered while using agile methods in education.

According to researchers from Texas University, it is very important to choose the proper project to be completed during a software engineering course[1]. This assumes that the project has the right size and complexity, so that the students can experience the necessity of using a software engineering method and succeed while using it too. A professor from Iowa University considers fulfilling the role of project owner the critical factor, because he/she seriously affects the success of the project by writing the user stories in a proper way, prioritizing them, keeping in contact with the customer and the project team as well[2]. Instead of one meeting per week, longer periods (for example: one-two weeks) assigned for team project work would be much more beneficial for students. To archive this, restructuring the semesters would probably be necessary which could result in a lot of pushback[3]. Teachers from Maryland University identify the need for initial training, where students learn about the agile values, develop their soft skills, so that they can apply them during the course [4]. Only this could assure that the power of agile methods would be manifested. All teachers agree, that using agile methods during software development courses is a current research problem and many aspects of it still need to be explored further.

Case studies

In 2016 six teachers from Eötvös Loránd University's Faculty of Informatics decided to use Scrum for a student-team project during a software engineering course.

There were three different courses, each of them adopted Scrum in a different way. There was a course, where the teacher fulfilled the role of Scrum master, product owner and customer as well. In another course the roles were assigned to several students. There were 2-week and 8-week long sprints, teams with 10 and teams with 3 members. As the assessment method, an interview-like test was used in one course and a complex evaluation schema was applied in another where students received points from one another too.

Teams have been analyzed along the Scrum roles, events and products. Another important group constitutes the solutions regarding education related questions like: How to choose the adequate software project to be realized? How to gain the most from the little time the students can work together? How to evaluate the students individually? etc.

Conclusions

I monitored three different courses two semesters long. At the end of the semesters I composed a questionnaire (based on international research papers, expanded with local specialties), and analyzed the results. Interviews were also taken (from teachers and students). International research conclusions where contracted with local conclusions.

This section presents the result of my qualitative empirical study, which is a list of good practices concerning all the important decisions regarding the use of Scrum in an academic course. The suggestions trace out a concrete method of applying Scrum at a software engineering course.

Scrum roles

It can give extra motivation to the students and make the simulation of software development more authentic if the project has a real customer. (This can be for example a teacher from another faculty of the university). The Scrum master and the project owner role can be fulfilled by students – this can assure that the respective ones get some valuable experience regarding Scrum roles. This is at the same time risky for the project team because the success of the software development process highly depends on these key roles. Therefore, it is recommended to use the right methods like interviews or tests, while choosing the appropriate students. It might be necessary to build a stronger relationship with the product owner student, because usually it is strange for them to stand for the interests of the customer rather than for interest of the (student) project team, to be strict about the acceptance criteria of user stories. The Scrum master student must get more attention from the teacher too. A separate short training regarding his/her role is recommended at the beginning of the semester. His/her attitude to the project can easily influence the attitude of all team members.

All students should be part of one Scrum team and the teams should be encouraged to get self-organized, independent, to pay attention to communication. Communication can be supported, by establishing an effective communication channel at the very beginning of the course; for example, beginning the course with a longer introduction cycle, ice breaker games, or team events. If the students meet once per week, I suggest creating maximum 5-member Scrum teams.

Scrum events

Three-week long sprints could be ideal considering that semesters are 13 weeks long. Choosing a longer sprint would not be profitable because the rare occurrence of demos and reviews. Using planning poker while planning the sprint can be very exciting for students. It is more like a game for them because they don't have enough experience in software development to archive a relevant estimation. But they can profit a lot from discussing the user stories repeatedly till a common estimation comes off. It is a time-consuming procedure, so it must be well prepared.

The students don't have the possibility to practice the daily stand up every day, so these must be realized during weekly meetings. It is important to keep the characteristics of the daily Scrum event so keep them short and compact. The original concept of the demo must be accomplished too: fluent, accurate, careful presentation of the software to the customer. The students enjoy learning different review techniques, diagrams and statistics. The teachers can benefit from the reviews even more: they get feedback from the students, can collect information about the teams, their needs, their concept about Scrum and agile values. It is very important to emphasize the team's strengths too, not only the weaknesses.

Scrum products

It is important to start the project with a good product vision. This can be written by the teacher or the students can practice writing it too. As far as the product backlog, the details are very important, especially the acceptance criteria. If we miss this at the very beginning it can cause a big disappointment for students later. The teachers must assist the project owner student because he/she may have little experience with user stories and this can cause trouble for the whole project team. The students have the tendency to forget about the Scrum table. As teachers, we may have to come up with different techniques to remind them of the Scrum table.

Education

In most cases the teachers cannot find out ahead of time what kind of technologies the students of the course are familiar with and at what level they are. Regardless, the teachers must choose a software to be assigned which presents a challenge, but a solvable one. One good practice is to opt for a software that can be realized in both a basic and a complex way to. Even if the focus of the course is the software product, it is a good idea to assign a period at the beginning of the course to get ready, to get to know the software development method. The positive effect of software development methods can only be expected if the team members know and apply the method correctly. This is especially true for agile methods.

The evaluation of the students is a critical factor too. If we put agile project management in the focus of a software development course, then we must evaluate students during the course to observe how they contrive the agile values too. This is hard to measure. Personal observation and the feedback of other team members (using anonymous questionnaires) can be used for it. We can reflect on different aspects in regard to a team member: communication, collaboration, adaptation to change, etc.

All in all, teachers and students agree that a 1,5 hours long class per week is too short for both team meetings and software development actions. The daily Scrum, communication, Scrum roles could be realized more effectively if the time dedicated for the class would be longer. At the same time, I consider that if students follow the good practices collected in this article in regard to Scrum that will significantly contributes to the success of software development projects. Considering other advantages as well [5], I recommend the use of agile methods in software engineering education.

Acknowledgements

The project has been supported by the European Union, co-financed by the European Social Fund (EFOP-3.6.3-VEKOP-16-2017-00002).

References

[1] K. M. C. C. S. L. Tucker Smith, "Software Engineering Senior Design Course: Experiences with Agile Game Development in a Capstone Project", ACM, 2011

[2] J. G. Kuhl, "Incorporation of Agile Development Methodology a Capstone Software Engineering Project Course", ASEE, 2014.

[3] F. M. Craig Anslow, "An Experience Report at Teaching a Group Based Agile Software Development Project Course", ACM, 2015.

[4] D. D. F. R. a. D. H. H. Sayani, "Use of Agile methods in software engineering education", IEEE, 2009.

[5] E. Ilyés, „Agilis szoftverfejlesztési gyakorlat megvalósításának lehetősége az ELTE Informatikai Karán”, 2017.

Towards Proper Differential Analysis of Static Analysis Engine Changes

Gábor Horváth, Réka Kovács, Péter Szécsi

Abstract: The design and implementation of heuristics for static analysis engines require detailed knowledge about the code to be analyzed. Extensive testing is therefore required to validate whether a change to the analysis engine is beneficial for real-world software projects.

The current practice of testing an analyzer on a fixed set of projects is not sufficient. Changes in the engine might affect language features that are utilized by only a fraction of the projects in the test suite. We explore a direction to ease the design of differential analysis experiments on a dynamic set of projects. This involves semi-automatic generation of the test set and evaluation of analysis results before and after applying changes to the engine. As the presented framework includes tools that aid the interpretation, reproduction, and sharing of analysis results, it might be valuable for a wide range of developers in the community.

Keywords: static analysis, symbolic execution, Clang, testing

Introduction

The proposition of a new patch to a static analysis engine involves disclosing information about the possible effects of the change. This normally includes analysis results on a few software projects before and after applying the patch.

Several problems arise during this process. Finding a set of test projects that truly show the effects of the patch can be a challenging task. Furthermore, a reviewer's request to extend the number of test projects might result in a significant amount of extra work for the patch author. This extra work is the result of the different components. With the continuous evolution of the static analysis engine, the author also needs to rerun the analysis on the old and the requested projets after a rebase. The results need to be processed so it can be easily digested by the reviewers. Ideally, the reproduction and extension of an analysis should be painless, and it should be possible to present results in an easily shareable and digestible format.

A related project, Corvig [1] is a tool to run dynamic and static analysis on projects and aggregate the results. Its emphasis is on collecting metrics about the analyzed projects and not on collecting metrics about the analyzers.

Overview

In this paper, we present a toolchain for the Clang Static Analyzer [2], a static analysis tool built on top of the Clang compiler for C family languages. The presented toolchain [3] aims to improve the situation by supporting both reviewers and authors in the following ways:

- help authors select a set of relevant projects for testing and run static analysis on them,
- aggregate statistics about the analysis (e.g.: how often a cut heuristic is triggered when building the symbolic execution graph),
- aggregate the results of the analysis,
- help authors and reviewers evaluate and share the results,
- help reviewers reproduce the results and maintain the tests.

The input of the toolset is a single, easy to interpret configuration file. The output is a HTML report with useful information and figures. The output also contains the input configuration for easier reproducibility.

Towards automatic test suite generation The conventional approach to testing is to run the analysis tool on a number of projects. Finding a sufficient amount of relevant real-world projects can be challenging. Ideal projects should be open-source for reproducibility and should exercise the right parts of the analyzer. For example, if the change is related to the modeling of dynamic type information, only projects using dynamic type information should be included. One possible option is to check a random sample of open-source projects, hoping to find enough that display all of the required traits. A slightly better approach is to use code searching and indexing services and for projects with interesting code snippets. These services, however, are optimized to present the individual snippets and suboptimal to retrieve the most relevant projects according to some criteria.

To mitigate this problem we present a script to harvest the results from an existing code search service and to recommend projects to be included in the test suite based on the results.

Towards easy reproduction and sharing The next problem is sharing results with reviewers. A regular pattern we see is the patch author sharing text files containing the results of the static analysis on certain projects. Text dumps of static analysis results are hard to interpret and the measurements are hard to reproduce. How did the author compile the project? Which version of the analyzed project was used? How did the author invoke the analyzer? What configuration options were used? What revision (commit) of the analyzer was used?

Our scripts use a concise configuration format that contains all the relevant information about the analyzed projects: repository, tag/commit, configuration options for the analysis, etc. Obtaining this configuration file enables reviewers to reproduce the exact same measurements at their convenience. They can also easily suggest modifications to the conducted experiment. Moreover, the results are not mere text dumps anymore but are presented on a convenient web user interface that also displays the path associated with the report. Other information such as the number of code lines of the project, version of the analyzer, analysis time, analysis coverage, and statistics from the analysis engine is recorded and figures like charts are generated automatically.

Towards more precise differential analysis Finally, the number of tools available to support differential analysis on a project is scarce. In case of the Clang Static Analyzer, we can only compare the number of bugs found, analyzer engine statistics, and the coverage percentage measured in basic blocks. All of these are aggregated scalar values missing positional information, with the statistics and the coverage being displayed individually for each translation unit.

We implemented analysis coverage measurement right in the heart of the analyzer. Using the extra information provided by the modified engine we can perform differential analysis on the coverage itself instead of settling for the study of coverage percentage values alone. Reviewers can check which new lines became covered after the change and which lines left the scope for some reason. We also support differential analysis of the analyzer reports.

Recommended workflow Using our toolset the recommended workflow is the following. The author of the patch provides reviewers with a link to the test results. Reviewers can choose to either merely look at the results or repeat the whole experiment based on the configuration, depending on the verification effort required for the change. They can also suggest changes to the configuration to gather more insight about the changes.

The Proposed Toolchain

Semi-automatic test suite generation Our script addressing the collection of relevant test project candidates for an engine change uses the `searchcode.com` service for its backend. For example, in order to test a new static analysis check written to detect `pthread_mutex_t` abuse, we might be interested in projects that use `pthread` extensively.

Using the following syntax we can specify the keywords to search for, the languages we are interested in, the desired number of projects:

```
1 python gen_project_list.py 'pthread_mutex_t' 'C C++' 5 ——output pthread.json
```

The above call creates a configuration file with the suggested projects in the following format:

```
{ "projects": [ { "url":"github.com/itkovian/torque.git", "name": "torque" }, ... ]
  }
```

Easy analysis reproduction and sharing The file above is almost enough to run the analysis on its own. The only extra information needed to be specified is the CodeChecker [4] server where analysis results are intended to be stored for later inspection.

```
{ "projects": ... , "CodeChecker": { "url" : "localhost:15010/Default" } }
```

CodeChecker is a tool designed to integrate the Clang Static Analyzer and Clang Tidy into C/C++ build systems. It also acts as a mature bug management system that supports the commenting on static analysis reports and the suppression of false positives. It has a convenient user interface to visualize the path of the path-sensitive bug reports and to support differential analysis. We can compare two analysis runs using CodeChecker to differentiate between common reports and those present only in a specific analysis run. CodeChecker's web GUI allows sharing the results with the rest of the world without the need of repeating the experiment. It can be used not only to share the bug reports, but the corresponding classifications and comments why some findings were false positives or true positives according to the author of the patch.

After this we are ready to run the analysis on the previously selected set of projects:

```
python run_experiments.py ––config pthread.json
```

The script checks out each of the projects, attempts to infer their build system and build them, runs the analysis and finally collects the results. At the time of writing this paper `autotools`, `CMake`, and `make` are supported as build systems.

In case a special build command is required or the build system is not yet supported, the user can specify the build command. Building a special version of the project characterized by a tag or a commit hash instead of top of tree is also possible and highly encouraged to get consistent results with subsequent experiments. Finally, differential analysis can currently be conducted by running the same projects multiple times with different options passed to the analyzer or using different versions of the analyzer. An example can be seen below.

```
  { "projects": [ ... ],
    "configurations": [
      { "name": "original", "clang_sa_args": "", },
      { "name": "variant A",
        "clang_sa_args": "argument to enable feature A",
        "clang_path": "path to clang variant" }
    ],
  ... }
```

A more precise differential analysis Currently, coverage measurements provided by the Clang Static Analyzer are limited. The engine can only record the percentage of basic blocks reached during the analysis of a translation unit, which is not sufficiently precise for multiple reasons. First, the analysis can stop in the middle of a basic block due to running out of the analysis budget for that specific execution path. Secondly, there is no precise way of merging information from different translation units. Finally, inline functions or templates in header files might appear in multiple translation units and their contribution will be counted multiple times upon attempting to aggregate information over translation units.

We implemented line-based coverage measurement based on the `gcov` format. We do not calculate coverage as an overall percentage value but record it separately for each line. This makes it possible to precisely aggregate coverage information over translation units. This also makes it possible to do differential analysis on the coverage itself. Our toolset includes scripts to aid that kind of analysis.

In some cases, we are interested in the reason behind a specific bug report disappearing when running the analysis with different parameters. Performing differential analysis on the coverage, we are able to determine whether the analyzer actually examined the code in question during both runs.

The Clang Static Analyzer can output different kinds of statistics such as the number of paths examined, the number of times a specific cut heuristic was used etc. Instead of having a fixed set of statistics

to collect we used some heuristics to process the output of the analyzer, in which we are able to automatically detect statistics and aggregate them over translation units.

We create an HTML summary of the statistics collected during the analysis. This report includes charts and histograms. After adding a new statistic to the analyzer engine the author only needs to add a single entry in the configuration file to make the toolset generate a figure based on that statistic.

Conclusions

We find the traditional practice of static analysis tool testing insufficient. One of the greatest problems is that a fixed set of test projects might not stress the newly introduced code paths of the analysis engine. The other concern is reproducibility, which is not only essential for reviewers tinkering with the measurements but also for any subsequent re-evaluation of the changes. Finally, the current practice of presenting the measurement results does not aid the interpretation of the raw data. Using an easier to digest presentation of the measurements could reduce the effort needed to evaluate the changes for the reviewers.

In order to mitigate these issues, we suggested a particular analysis workflow and developed a supporting toolchain the Clang Static Analyzer. These tools not only help to collect relevant candidate projects for testing but also perform a proper differential analysis on the test projects and generate easy to interpret figures for the reviewers. We also added a new line-based coverage measurement mechanism to the Clang Static Analyzer engine to improve the precision of the analysis.

Acknowledgement

Supported by ELTE Eötvös Loránd University in the frame of the ÚNKP-17-3 New National Excellence Program of the Ministry of Human Capacities.

References

[1] P. Marinescu, P. Hosek, and C. Cadar, "Covrig: A framework for the analysis of code, test, and coverage evolution in real software," in *Proceedings of the 2014 International Symposium on Software Testing and Analysis*, ser. ISSTA 2014. New York, NY, USA: ACM, 2014, pp. 93–104. [Online]. Available:

[2] "Clang static analyzer," 2018. [Online]. Available: https://clang-analyzer.llvm.org/

[3] G. Horvath, R. Kovacs, and P. Szecsi, "Clang static analyzer testbench," 2018. [Online]. Available: https://github.com/Xazax-hun/csa-testbench

[4] "Codechecker," 2018. [Online]. Available: https://github.com/Ericsson/codechecker

Projection selection with sequential selection methods using different evaluation measures

Gábor Lékó, Péter Balázs, László G. Varga

In binary tomography [1] the goal is to reconstruct the inner structure of homogeneous objects from a low number of their projections. There is a strong connection between the quality of a binary tomographic reconstruction and the choice of angles of the projections [2]. For selecting proper projection directions different evaluation values can be used. If the blueprint data is given we can calculate the RME (Relative Mean Error) value of a reconstruction and use it to identify informative angles. In the absence of blueprint data, we can calculate the uncertainty. In many cases there can be several solutions of the reconstruction problem. Knowing all the reconstructions we could calculate the probability of a single pixel taking the value 1. Given the probabilities we can determine the uncertainty of a reconstruction [3].

We provide different projection selection algorithms based on sequential selection methods [4] in order to find the "most informative" projection set. We also compare them with already existing algorithms. For the optimization the two abovementioned evaluation values were used. To show that uncertainty can be useful in projection selection with a great confidence and to compare the performance of the given algorithms we performed experimental tests on a set of binary software phantoms.

Acknowledgements

This research was supported by the project "Integrated program for training new generation of scientists in the fields of computer science", no EFOP-3.6.3-VEKOP-16-2017-0002. The project has been supported by the European Union and co-funded by the European Social Fund.

References

[1] G.T. Herman. Image Reconstruction from Projections. Fundamentals of Computerized Tomography, second ed., Springer-Verlag, London, 2009.

[2] L. G. Varga, P. Balázs, A. Nagy. Direction-dependency of binary tomographic reconstruction algorithms. Graphical Models, 73(6):365-375, 2011. Computational Modeling in Imaging Sciences.

[3] L. G. Varga, L. G. Nyúl, A. Nagy, P. Balázs. Local and global uncertainty in binary tomographic reconstruction. Computer Vision and Image Understanding, 129 52-62 (2014).

[4] P. Pudil, J. Novovičová, J. Kittler. Floating Search Methods in Feature Selection. Pattern Recognition Letters, 15(11):1119-1125, 1994.

Axiom-based property verification for P4 programs

Gabriella Tóth, Máté Tejfel

Abstract: We produce an axiom-based program properties verification method for P4 programs. P4 is a special, domain specific, declarative programming language to develop network packet forwarding. P4 is quite new, and different from general-purpose programming languages so it is an important idea to understand its behavior and to verify it. The operational semantics of P4 is reachable in $\mathbb{K}$ framework, so there is an opportunity to do its verification based on its operational semantics, but it is a low level solution, therefore the proof of complex properties can be too difficult and costly. So we would like to verify the program in a higher abstraction level, in which we would introduce axioms, which are correct in the operational semantics. Using these axioms we can create easier and more transparent proof.

Keywords: P4, $\mathbb{K}$ framework, verification, operational semantics

Introduction

This paper analyze P4 programs and program properties. We chose P4, because it is a new, special, domain specific, declarative programming language, which is created for developing network packet forwarding. P4 has a reachable operational semantics in $\mathbb{K}$ framework, so we can start the verification based on these semantics rules. But the disadvantage of this way is that, it is a low level solution, therefore the proof of complex properties can be too difficult and costly. Consequently, we would like to create a higher abstraction level with introducing operational semantics based axioms, to make an easier way for proof and to reach proof's end sooner.
The above mentioned $\mathbb{K}$ framework is an executable semantic framework, in which we can add any program language's syntax and operational semantics for verify programs with symbolic execution [1][2]. The $\mathbb{K}$ uses configurations to follow the program states, and we can give the semantics rules as a relation from configurations to configuration. The P4 operational semantics is given in this way, so our axioms will be developed in the same style. Using these axioms we can produce an axiom-based program property verification method for P4 programs.

The P4 language

P4 language is a new, domain specific, declarative programming language, which is created for a target independent, protocol independent and reconfigurable way to develop network packet forwarding [3][4]. The board members of the P4 language consortium come from such a famous institutions as Stanford University, Princeton University and Google. So far, there are two version of the language, the P4_{14} and the P4_{16}. P4_{14} has a available operational semantics in $\mathbb{K}$ framework [5], so in the paper this version will be used.

An example for P4 program

In Figure 1. we can see a simple P4 program. This example code snippet demonstrated those language elements of P4 that are essential for understanding our results presented in this paper. To give the types of headers is an important part to handle the packets. We can create these elements as a $header\_type$, with the name and contained fields and the bit-width of fields. With these header types we can define header instances, with which we can handle the real headers in runtime. In our example these elements appear in the first and the third line. In the first line one header type is defined, which name is $simple\_header\_t$, and has 3 field, the $field\_a$ and $filed\_c$ with 8 bit-width and the $field\_b$ with 16 bit-width. This type has an instance, with $simple\_header$ name (line 3).

We need to have parser in our programs, which gives how the first part of the packet have to be divided into headers. In the example the parser (lines 5-8) unwraps the header $simple\_header$.

The actions contain the operations which can be executed in the headers. There are predefined primitive actions, and the developer can also compose new actions build up from sequences of primitive actions. In Figure 1. there is only one action named $simple\_action$, which calls one $modify\_field$. The

```
1  header_type simple_header_t { fields { field_a : 8; field_b : 16; field_c : 8; } }
2
3  header simple_header_t simple_header;
4
5  parser start {
6      extract(simple_header);
7      return ingress;
8  }
9
10 action simple_action() { modify_field(simple_header.field_b, 3); }
11
12 table simple_table {
13     reads { simple_header.field_a : exact; }
14     actions { simple_action;}
15 }
16
17 control ingress { apply(simple_table); }
```

Figure 1: A simple P4 input file

modify_field is a primitive action, and it tries to change the first parameter's value for the second parameter, so here, we try to change the value of the field named *field_b* of the header named *simple_header* to 3.

The declarations of P4 'match+action' tables define what actions can execute based on the processed packet's values. This element has a name, a 'reads' and an 'actions' part. In the 'reads' we can show those header instance's field, which we would like to use during matching, and the type of matching. In the 'actions' part there are the set of executable actions. In our simple program, there is one 'match+action' table (lines 12-15) as *simple_table*, and it would like to match the *simple_header*'s *field_a* field in the exact way, and it can execute one action, the above mentioned *simple_action*.

After the analysis of the packet, the next step is the execution of the control function starting with the ingress control function. These functions define the sequence of the tables' execution for the packet, and define how to continue the execution after a matching. In our simple program there is only one ingress control function, which calls our table's execution in the most simple way.

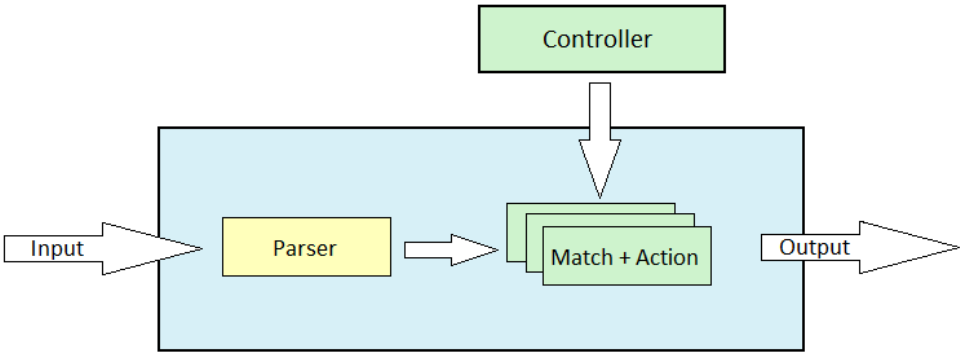


Figure 2: A simplified illustration for P4 behavior

The P4 programs' behavior

P4 has a different behavior as other common programming languages. P4 programs contain only declarations about the packets' headers, parsers, actions, etc. In Figure 2, we can see a simplified illustration for its conduct. First, the packets go through the parser and that creates a set, which contains valid header instances for the packet. Initially 'match+action' tables are empty, just those information is given, which is in the declaration. There is an external controller, which fills these tables with concrete

pairs. One pair contains a matching value and the action, which have to be executed when the matching is fine. The uploaded 'match+action' tables operate on parser made set, and this process is based on the control functions. After all processing finished, the resulting packet will be transmitted.

Motivation

A simple way for the verification, if we analyze concrete P4 programs, like a $modify\_field$ effect in a given value of a header instance's field. For example in Figure 1, there is only one action, which modify the value of $field\_b$ to 3 (line 10). We could examine the value's changing. If in the initial state $field\_b$ has any given value, then in the final state its value will be the same or 3. A program property, where we work with concrete values, and concrete elements, can be proved easily using the operational semantics.

In the next step, we can try to go further, and generalize the property. If initially we have a given field with any value and a program contains one $modify\_field$ action for this field with a new value, then the program would go to a final state, where the value of the given field is the new one. Using operational semantics's rules it would be a complex proof, because it would work with meta-variables over the value and field, with which more calculations of inside steps would not be obvious. There is another properties, which could be examined, if initially we have a given field and a program contains only one $modify\_field$ action with another name of fields, then the program would go to a final state, where the value of the given field will be the same. It seems a similar statement, but its proof is really hard using only the operational semantics's rules.

In $\mathbb{K}$ framework for the symbolic execution we need to give a file, which contains the behavior of controller, so we need to give the concrete filling of the 'match+action' tables. In the proofs of previous properties, which uses only operational semantics, there would be rules which based on the controller's behavior. The controller can only call actions, which were declared in the used P4 program, therefore an even further way to say statements about a P4 program is that if we do not deal with the controller, and say properties only about P4 declarations. For instance, a statement could say, if we have actions, which contains only $modify\_field$ primitive actions, and there is a field, which is not appear in any first parameter of $modify_field$s, then the value of this field won't change after any execution, it does not matter how the tables looks like, and what type of actions will be run, because we do not have any action, which would change its value. In this point, we need to go a higher abstraction level, because these type of statements cannot be proved using only operational semantics. In this level instead of symbolic execution, our prove based on some operational semantics's rules and axioms, or in an even higher level only axioms with deduction rules. In this way it can be an axiomatic semantics [6]. The introduced axioms will be defined and proved based on the operational semantics, and sometimes the operational semantics's structure is not enough for the proof, so we will supplement the rules and configurations.

The 'Unchanged Field' axiom

In the example we analyze the property, when the value of examined field is unchanged. Using only the operational semantics's rules, the proof of this property is complicated, so we introduce an axiom for easier proof, which contains conditions in the upper part, and a Hoare triple in the lower part:

$$\frac{f \in instanceFields(S) \text{ and } f \notin modifiedFields(S)}{\{f = X\}\ S\ \{f = X\}}$$

In the axiom, S is the examined program, X indicates any value of a field, $instanceFields(S)$ is a set of fields, which appear in the program and $modifiedFields(S)$ is a set of fields, which value can be changed during the program run. The $modifiedFields$ contains those fields, which appear in the first parameter of any primitive actions, because a field's value can be changed only in this way. A field is identified by the instance, which contains it, so an element of the two sets has $instanceName.fieldName$ format.

The given axiom expresses that, if there is a field in the program and it is not in the $modifiedFields$ set, after the program run, its value will be the same as in the initial state.

Using this axiom in the case of the program introduced by Figure 1 (sign it with S1), we can easily prove the stability of $simple\_header.field\_a$ and $simple\_header.field\_c$. The two sets are computable:

$instanceFields(S1) = \{simple\_header.field\_a, simple\_header.field\_b, simple\_header.field\_c\}$

$modifiedFields(S1) = \{simple\_header.field\_b\}$.

The axiom's conditions are true for $simple\_header.field\_a$ and $simple\_header.field\_c$, so these fields will not change during the execution of $S1$ program.

Acknowledgements

This work has been supported by the European Union, co-financed by the European Social Fund (EFOP-3.6.3-VEKOP-16-2017-00002).

References

[1] A. Ştefănescu, D. Park, S. Yuwen, Y. Li, G. Roşu: Semantic-Based Program Verifiers for All Languages. In *Proceedings of the 2016 ACM SIGPLAN International Conference on Object-Oriented Programming, Systems, Languages, and Applications* (OOPSLA'16), pages 74-91. ACM, Netherlands, 2016

[2] G. Roşu: $\mathbb{K}$: A Semantic Framework for Programming Languages and Formal Analysis Tools. In *Dependable Software Systems Engineering*, pages 186-206. IOS Press, Netherlands, 2017

[3] The P4 Language Consortium: The P4 Language Specification *https://p4.org/p4-spec/p4-14/v1.0.4/tex/p4.pdf*, 2017

[4] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese, D. Walker: P4: Programming Protocol-Independent Packet Processors. In *ACM SIGCOMM Computer Communication Review*, pages 87-95, ACM, USA, 2014

[5] The P4 Language's operational semantics in $\mathbb{K}$ framework: https://github.com/kframework/p4-semantics

[6] R. D. Cameron: Axiomatic Semantics of Programming Languages. 2002

Semi-Supervised Training of Cell-Classifier Neural Networks

Gergely Pap, Tamás Grósz, László Tóth

Abstract: Nowadays, microscopes used in biological research produce a huge amount of image data. Manually processing the images is a very time-consuming and resource-heavy task, so the development and implementation of new automatic systems is required. Moreover, as we have access to a large amount of unlabeled data, while labels are only available for a small subset, these novel methods should be able to process large amounts of unlabeled data with minimal manual supervision. Here, we apply neural networks to classify cells present in biological images, and show that their accuracy can be improved by using semi-supervised training algorithms.

Keywords: DNN, RBM, semi-supervised learning

Introduction

The dataset used in this paper consists of features extracted from cells based on image processing considerations and biological requirements. The final goal is to automatically classify the cells by their properties regarding their gene-expression profile and other important cellular descriptors. As only a small subset of the database was annotated manually, here we seek training algorithms that allow us to exploit the unlabeled data to improve classification accuracy.

The processing of large datasets is a task which requires considerable manual labor and time, so usage of an automatic system is widely recommended. Artificial Neural Networks (ANNs) have made significant progress in the past few years within a wide area of scientific fields (for example: image processing, voice recognition, machine translation etc.). Neural networks generally require an adequately large annotated training database before they can be used to classify previously unseen or new data. However, creating an annotated database of considerable size is a labor-heavy and costly process that requires tedious manual work. Hence, it would be optimal if human supervision could be minimalised during the creation of the training data, moreover, if unlabeled data could be used to improve classification accuracy. The above-mentioned goals are achievable using Restricted Boltzmann Machines (RBMs) [2]. In the case of RBMs, an unsupervised learning method can be applied called Contrastive Divergence (CD), which pretrains the network using unlabeled data. Then, the subsequent regular supervised learning step using the labeled data usually attains better classification accuracies than the ones it achieved without pretraining.

During our research we used conventional "shallow" and deep artificial neural networks as a baseline trained via labeled data and then we compared their performance with two RBMs, which were configured to apply different learning algorithms (a generative and a discriminative one) using the unlabeled data set.

Data

The data used in our study comes from images of cells in different biological states. The cells were captured using fluorescent microscopy and 254 features based on biological properties (regarding gene-expression and other cellular descriptors) were then extracted using image processing. Biologists then manually labeled the cells in a small subset of the data, sorting them into 12 different classes. There are roughly 1,8 million entities in the dataset and only 2600 were labeled by hand. Classes included cells separated by their membrane features (normal, ruptured, vesical), by their place in the cells' life-cycle (normal, dying, apoptotic) and by their visual properties in connection to the microscopical and image processing traits (normal, segmentation error and well-defined contour). The features included the position of the cell, distance from other cells, properties of the different colour channels, biological attributes of cellular organs (e.g. membrane, nucleus, cytoplasm).

We split the labeled data into three subsets: 2100 were used for supervised training, 300 for validation (also known as development dataset) and 200 for testing.

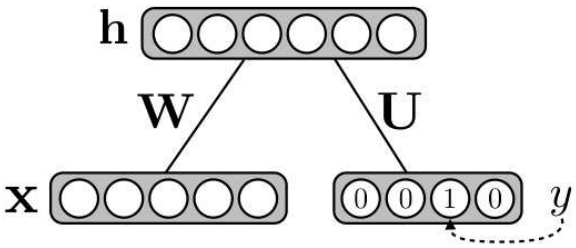


Figure 1: ClassRBM structure where x is the input vector, h denotes the units of the hidden layer and y represents the labels. Furthermore, W and U are the respective weight matrices.

Methods

Restricted Boltzmann Machines are often used in machine learning problems as feature extractors for another learning algorithm or as a good initialization strategy for deep neural networks. However, it has been shown that RBMs can also be utilized as a standalone discriminative classification method. In this work, Classification Restricted Boltzmann Machines (ClassRBMs) were pretrained using unlabeled data to improve classification accuracy.

RBMs have two layers of neurons (hidden and visible), where connections within the layers are forbidden. The information extracted by the hidden layer tries to model the joint distribution of the input and the hidden states, passing the data on to the next set of layers. Though standard RBMs are trained only to model the inputs of a task, the advantage of ClassRBMs is that they can also model the joint distribution of the inputs and the corresponding target classes, as Figure 1 illustrates. Traditionally, training is finished by converting the RBM into a regular feed-forward network which is trained on labeled data using backpropagation [3].

Classification Restricted Boltzmann Machines

In a generative setting, the RBM uses an energy function that models the joint distribution of every possible visible and hidden vector pair

$$p(v,h) = \frac{1}{Z} e^{-E(v,h)}, \tag{17}$$

where the denominator Z converts the result of equation (1) into a probability, it is calculated as the summation of all possible hidden and visible vector pairs as follows

$$Z = \sum_{v,h} e^{-E(v,h)}. \tag{18}$$

The RBMs energy function is given by

$$E(v,h) = -b^T v - c^T h - v^T W h \tag{19}$$

where b is the bias of the visible layer and c is the bias of the hidden layer [1].
We then can calculate an estimate of the gradient regarding the model parameters for any $\theta \in \Theta$

$$\frac{\delta logp(y_i, x_i)}{\delta \theta} = -E_{h|y_i,x_i}[\frac{\delta}{\delta\theta} E_{y_i,x_i,h}] + E_{y,x,h}[\frac{\delta}{\delta\theta} E_{y,x,h}]. \tag{20}$$

As we were interested in the prediction of the correct labels, we used Discriminative Restricted Boltzmann Machines to model $p(y|x)$ instead of $p(y,x)$

$$L_{(disc)}(D_{train}) = -\sum_{i=1}^{|D_{train}|} logp(y_i|x_i). \tag{21}$$

The RBMs minimizing the function $L_{(disc)}$ are called Discriminative Restricted Boltzmann Machines (DRBMs). An RBM containing enough layers can be considered as an universal approximator for binary

inputs, which means that DRBMs are also universal approximators for the joint distribution of binary inputs and class labels y.

DRBMs use an algorithm called Contrastive Divergence (CD) for training, however in the model $p(y|x)$ can be calculated directly, we can then compute the exact gradient using Algorithm 1:

Algorithm 1 Contrastive Divergence for discriminative RBM training

1: INPUT: training pair (y_i, x_i) and learning rate λ
2: a $\leftarrow$ b means a is set to the value of b
3: a $\sim$ b means a is sampled from p
4: Positive Phase
5: $y^0 \leftarrow y_i, x^0 \leftarrow x_i, \hat{h}^0 \leftarrow sigm(c + Wx^0 + U\overrightarrow{y}^0)$
6: Negative Phase
7: $h^0 \sim p(h|y^0, x^0), y^1 \sim p(y|h^0), x^1 \sim p(x|h^0)$
8: $\hat{h}^1 \leftarrow sigm(c + Wx^1 + U\overrightarrow{y}^1)$
9: **Update**:
10: **for** $\theta \in \Theta$ **do**
11: $\quad \theta \leftarrow \theta - \lambda(\frac{\delta}{\delta\theta}E(y^0, x^0, \hat{h}^0 - (\frac{\delta}{\delta\theta}E(y^1, x^1, \hat{h}^1)$
12: **end for**

A semi-supervised approach introduces constraints to the model, in this case the decision surface of the model is modified by the requirements imposed upon the RBM, as it tries to approximate a good generative model of the unlabeled data.

For our experiments we used an RBM implemented in MATLAB. This toolbox also contained (an ANN) implementation and the code to execute both discriminative and generative training of RBMs in a semi-supervised setting [4].

Results

To have a baseline result first we used a DNN with 3 hidden layers each containing 500 nodes with sigmoid activation function. The network was trained using the standard backpropagation algorithm. The DNN was trained for 60 epochs with 0.1 learning rate. The baseline's classification accuracy was around 67% on average. One classRBM had 100 neurons in three layers with 0.01 learning rate, the other 500 neurons in one layer and a learning rate of 0.05. The classRBM trained only on the labeled data performed a little better, averaging around 74%. However when the discriminative classRBM was used in conjunction with the unlabeled data in a semi-supervised learning scenario, the numbers of the correctly classified cells increased to 85%. After experimenting with the parameters of the training, we found that the values presented here yielded the best results in each particular case.

Conclusions

Here, we studied machine learning methods that allow us to make use of unlabeled data, besides the conventional training step that requires labeled data. We achieved improvements in classification accuracy by using large amounts of unannotated data for pretraining the network with a discriminative approach. After pretraining, training was finalized using a smaller set of labeled data. After the comparison of the results produced by the generative and discriminative methods, we experienced better performance in case of the latter classRBM. Altogether we found that the accuracy of DNNs can be improved by using semi-supervised training algorithms.

Method	Hid. Structure	Epochs	Learning Rate	Dev	Test
Baseline ANN 1	3x100	45	0.5	68%	65%
Baseline ANN 2	3x500	60	0.1	70%	67%
Gen. classRBM 1	3x100	45	0.01	68%	71%
Gen. classRBM 2	500	60	0.05	73%	69%
Disc. classRBM 1	3x100	45	0.01	77%	73%
Disc. classRBM 2	500	60	0.05	79%	83%
Semi-sup. Gen. classRBM 1	3x100	45	0.05	67%	68%
Semi-sup. Gen. classRBM 2	500	60	0.05	69%	64%
Semi-sup. Disc. classRBM 1	3x100	45	0.05	79%	78%
Semi-sup. Disc. classRBM 2	500	60	0.01	88%	85%

Table 1: Results

References

[1] Srikanth Cherla, Son N. Tran, Tillman Weyde, and Artur S. d'Avila Garcez. Generalising the Discriminative Restricted Boltzmann Machine. *CoRR*, abs/1604.01806, 2016.

[2] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[3] Hugo Larochelle and Yoshua Bengio. Classification using Discriminative Restricted Boltzmann Machines. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 536–543, New York, NY, USA, 2008. ACM.

[4] Soren Kaae Sonderby. RBM toolbox is a MATLAB toolbox for online training of RBM and stacked RBM's. 2014.

Feature extraction and classification for pupillary images of rats

György Kalmár, Alexandra Büki, Gabriella Kékesi, Gyöngyi Horváth, László G. Nyúl

Abstract: The investigation of the pupillary light reflex (PLR) is a well-known method to provide information about the functionality of the autonomic nervous system. Pupillometry, a non-invasive technique, was applied in our lab to study the schizophrenia-related PLR alterations in a new selectively bred rat substrain, named WISKET. The pupil responses to light impulses were recorded with an infrared camera; the videos were automatically processed and features were extracted. Besides the classical statistical analysis (ANOVA), feature selection and classification were applied to reveal the significant differences in the PLR parameters between the control and WISKET animals.

Keywords: schizophrenia, pupillometry, classification, curve properties

Introduction

Patients with schizophrenia, besides the well-known behavioral symptoms, also show autonomic dysregulation, including impaired pupillary function, which is a sensitive and reliable source of information about the function of the nervous system [1].

Pupillometry is a simple, non-invasive technique for the assessment of the autonomic nervous system function by testing the pupillary light reflex (PLR), meaning the contraction of the pupil in response to light. During the test, the changes of the pupil diameter are recorded and the size of the pupil is measured offline in each video frame.

Our study investigates the PLR to reveal the schizophrenia-related autonomic alterations in WISKET rats. Related works are summarized in Section 2. The novelties of the measurement and feature extraction methods are described in Section 3 and in Section 4, respectively. Section 5 shows the steps of data analysis. In Section 6 the results of the analysis are presented. Possible directions of future work are mentioned in Section 7 and Section 8 includes a short summary.

Related works

Developing reliable and predictive animal models for any complex psychiatric disorders, such as schizophrenia, is essential to improve our understanding of the neurobiological basis of the disorder. Recently, a new rat model of schizophrenia has been developed, named WISKET [5, 6, 8, 11].

Clinical studies in schizophrenic patients using pupillometry revealed impaired autonomic regulation [1, 2, 4, 9, 12]. PLR was also investigated in rodents [10]; however, no data are available from animal models of neuropsychiatric disorders, including schizophrenia.

Measurement process

Two series of experiments were performed in sedated (n=54) or anesthetized (n=20) male control Wistar, and WISKET rats. After a 10-minute dark adaptation period, the recording lasted for 15s in sedated, and for 60s in anesthetized animals. The animals were positioned in front of a camera, and exposed to an intensive light stimulus (approx. 300cd/m^2 for 600ms) into the left eye. The digital camera recorded pupillary responses at a speed of 24 frames-per-second under infrared illumination.

During the PLR measurements, the rats show several minor movements, which can affect the quality of the recorded videos. Furthermore, these albino rats lack pigments in their body including their eyes, reducing the contrast between the iris and the pupil. Specifically designed pupil detection and measurement algorithms were used to handle these quality drawbacks [7]. The input of the algorithm is a video recording and the output is a curve based on the measured pupil diameters in each frame.

Feature extraction

To compare the responses of the animals, we extracted well-describing features from the PLR curves, which are relevant from the pathophysiological point of view and are suitable to emphasize the differences between the groups regarding the autonomic nervous system activities.

An automated feature extraction method was designed, which produces 40 features. Many of them are basic, traditional parameters like initial diameter, which is the size of the pupil before the light impulse; minimum diameter; reaction time; maximum redilated diameter; etc. We also implemented new features to obtain information about the dynamics of the response, thus 11 velocity related descriptors were introduced. For example, the average and the maximal contraction velocities were determined and the time required to reach the latter was measured. The velocities at different points of the redilation phase were also calculated.

We introduced novel, smoothness related descriptors, as well. A polynomial curve with a given order was fitted on the redilation part of the response. The area between the original and fitted curves serves as a measure of non-smoothness. We chose 5^{th} order polynomials, which were flexible enough to follow the slow perturbations of the original curve and indicated only the short, abnormal swings. In Figure 1, a representative response curve and a marked subset of the extracted features is presented.

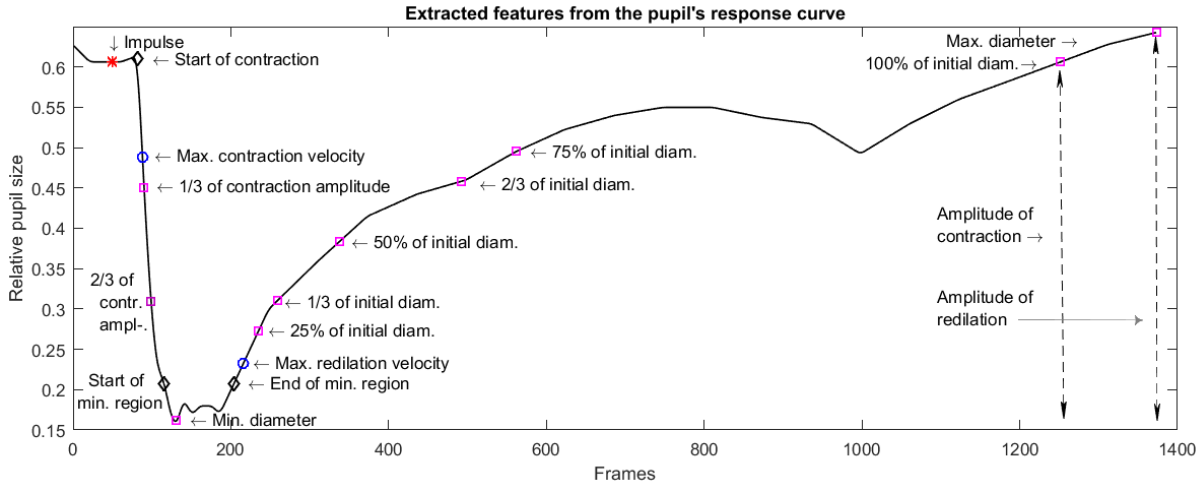


Figure 1: Pupillary light response curve and a marked subset of the 40 features.

Analysis

The sedated and the anesthetized animals were analyzed separately.

One-way ANOVA was used for the analysis of the extracted features. The relationships between pupil parameters were assessed by linear regression analysis and calculation of the Pearson correlation coefficient. Only probabilities lower than 0.05 were considered significant.

Besides the investigation of the differences in PLR parameters, the other goal is to use pupillometry as a quick examination to help classify the animals during the breeding process. Therefore, we trained a binary decision tree to investigate the possibility of classification and evaluated the model by using cross validation.

Results

The detailed discussion of the results of statistical analysis can be found in our recent study [3]. The trained decision tree selected almost the same features as predictor variables as the results of statistical analysis suggested.

In the sedated group, which has the bigger cardinality, the control and test animals show differences in many features. The initial and minimum pupil diameters are larger and the degree of the constriction

was lower in the WISKET rats. The flatness of the curve (length of the minimum region) and the contraction time were shorter in the control group. These results are in accordance with previous studies mentioned in Section 2. The decision tree achieved 71% accuracy measured by cross-validation. The algorithm selected the following predictors: minimum diameter; initial diameter; contraction time; average redilation speed.

The analysis showed that the group of anesthetized animals cannot be divided into two classes. We assume that while anesthesia can prevent the stress and allows a convenient investigation of pupillary reactions for a longer period, it also diminishes the autonomic responses. The classifier achieved only 60% accuracy; the selected predictors were: amplitude of contraction; average redilation speed; time required to reach the maximal redilation speed.

Future work

As the number of animals is limited and the cost of one sample is high, we are reconsidering the measurement process, extending the recording time and stimulating the rats with different light sequences. Thus, with this method more features could be extracted and new correlations between PLR parameters might be explored. Our further purpose is to develop a more robust recording process. We have already designed a pilot setup, in which an IR LED ring is applied to illuminate the eye and to induce a "bright pupil effect" resulted by the reflection of light from the retina.

Summary

We applied pupillometry to study how the schizophrenia-like alterations affect the PLR in the WISKET rats. We assigned a 40-dimensional feature vector to each recorded video and analyzed the dataset with statistical methods, and also trained a decision tree based classifier. The results demonstrate that pupillary control shows significant alterations in WISKET rats, and classification based on pupillometry might be applied as an additional examination during the breeding process.

References

[1] K.-J. Bär, M. K. Boettger, S. Schulz, C. Harzendorf, M. W. Agelink, V. K. Yeragani, P. Chokka, and A. Voss. The interaction between pupil function and cardiovascular regulation in patients with acute schizophrenia. *Clinical Neurophysiology*, 119(10):2209–2213, 2008.

[2] K.-J. Bär, M. Koschke, M. K. Boettger, S. Berger, A. Kabisch, H. Sauer, A. Voss, and V. K. Yeragani. Acute psychosis leads to increased qt variability in patients suffering from schizophrenia. *Schizophrenia research*, 95(1):115–123, 2007.

[3] A. Büki, G. Kalmár, G. Kékesi, L. G. Nyúl, and G. Horváth. Impaired pupillary control in "schizophrenia-like" WISKET rats. *Autonomic Neuroscience: Basic and Clinical*, 2018. Under revision.

[4] H. Hermesh, R. Shiloh, Y. Epstein, H. Manaim, A. Weizman, and H. Munitz. Heat intolerance in patients with chronic schizophrenia maintained with antipsychotic drugs. *American Journal of Psychiatry*, 157(8):1327–1329, 2000.

[5] G. Horváth, P. Liszli, G. Kékesi, A. Büki, and G. Benedek. Characterization of exploratory activity and learning ability of healthy and 'schizophrenia-like' rats in a square corridor system (ambitus). *Physiology & behavior*, 169:155–164, 2017.

[6] G. Horvath, Z. Petrovszki, G. Kekesi, G. Tuboly, B. Bodosi, J. Horvath, P. Gombkötő, G. Benedek, and A. Nagy. Electrophysiological alterations in a complex rat model of schizophrenia. *Behavioural brain research*, 307:65–72, 2016.

[7] G. Kalmár, A. Büki, G. Kékesi, G. Horváth, and L. G. Nyúl. Image processing-based automatic pupillometry on infrared videos. *Acta Cybernetica*, 23(2):599–613, 2017.

[8] G. Kékesi, Z. Petrovszki, G. Benedek, and G. Horváth. Sex-specific alterations in behavioral and cognitive functions in a "three hit" animal model of schizophrenia. *Behavioural brain research*, 284:85–93, 2015.

[9] A. Lidsky, G. Hakerem, and S. Sutton. Pupillary reactions to single light pulses in psychiatric patients and normals. *Journal of Nervous and Mental Disease*, 1971.

[10] K. Mohan, M. M. Harper, H. Kecova, E.-A. Ye, T. Lazic, D. S. Sakaguchi, R. H. Kardon, and S. D. Grozdanic. Characterization of structure and function of the mouse retina using pattern electroretinography, pupil light reflex, and optical coherence tomography. *Veterinary ophthalmology*, 15(s2):94–104, 2012.

[11] Z. Petrovszki, G. Adam, G. Tuboly, G. Kekesi, G. Benedek, S. Keri, and G. Horvath. Characterization of gene–environment interactions by behavioral profiling of selectively bred rats: The effect of NMDA receptor inhibition and social isolation. *Behavioural brain research*, 240:134–145, 2013.

[12] T. P. Zahn and D. Pickar. Autonomic activity in relation to symptom ratings and reaction time in unmedicated patients with schizophrenia. *Schizophrenia research*, 79(2):257–270, 2005.

Software as a Service operation model in cloud based ERP systems

István Orosz, Attila Selmeci

Abstract: Cloud based operation model has become a new standard in ERP system implementation. This has brought a new software abstraction layer over the already existing datacentre model, which covered all the middleware, operating system and database level architecture elements. This new abstraction layer covers the complexity of the middleware and below layers, making it possible for the owners to deal with only implementing the core business logic. There is always a risk that more resources are allocated to the infrastructure side, using this model resources can be reallocated back to the business side. Overall this could lead to longer software lifecycle, because the core logic is separated from the rapidly changing implementation layer. The standard release management, which covered the way from pre alpha state to the gold release, was substituted in this model with a continuous release process. This leads to the possibility of using the latest version of the software always in the Software as a Service operating model. The code reusability (and refactoring) is also modified, having an abstraction layer between the platform independent model and the platform specific model. The transformation layer between these two has to be able to ensure the long term reliability of the software product. Having a proper transformation layer keeps the roles separated, the end user do not have to build up an IT support background, because this layer hides the infrastructure questions. This paper focuses on the change management procedures when change the operation environment from the standard datacentre solution to the cloud based SaaS model.

This paper describes the different directions which can be used for an ERP environment offered by cloud technology, and focuses on the diverse change management procedures on switching the operation environment from the standard on premise datacenter solution to the cloud based SaaS model.

Keywords: cloud base operation model, ERP, SaaS, model driven architecture, change management, code reuse

Introduction

Nowadays the cloud-based solutions are very popular, but not always understood. Many companies would like to use it, but do not know how to start, for what use it in the current solutions. There are several option offered by ERP software vendors and cloud providers to own bigger market share. If we consider the ERP offerings, many services were developed originally for cloud, some others are enabled to be provided on cloud, not only as on premise solutions. There are some efforts to rebuild existing, proven solutions for cloud usage as well.

Standard development and release strategy can be grouped in different categories, like: Diagnostic, Analysis, Design, Development, Deployment and Operation [1]. In the datacentre based operational model, this required the close interaction of the organization, for the whole software lifecycle. When migrating or upgrading software solutions for cloud based operation, these phases will act differently.

The cloud-based operation cannot be separated from the provided services. The two of them act together as a new abstraction layer for software solution, where layers responsible for the implementation are strictly segregated from layers, which are responsible for modeling the core business logic.

The article is structured as follows. The first section gives a definition for the cloud based ERP operation models focusing on Software as a Service and then gives a brief review on the impact of information technology applications. Then the importance of code reusability (and refactoring) is demonstrated through an available tool in Microsoft Dynamics AX environment, for checking the integrity and consistency of code relevant changes and change management. On the other hand the popular SAP solutions are also mentioned by examples and implementation difficulties.

Service based operation in cloud environments

The rest of the paper describes the work that has been done. The expected length of the paper (with the references included) is four pages, please fill, but do not exceed this.

There are more service based operation models in cloud environment, according to what kind of service level is hidden from the business. There are three main operation models [2]:

- **Infrastructure as a Service (IaaS):** provides the lowest set of hosting services. The service level takes care of the virtualization, servers, storage and networking, so almost the infrastructure-like operations.
- **Platform as a Service (PaaS):** one step over the IaaS, this level additionally handles the operating system, middleware interfaces and the runtime software modules.
- **Software as a Service (SaaS):** over the PaaS, the customer has to deal with modellng the business processes implemented by the software solution, and can use the whole underlying infrastructure as a cloud service.

The services can be offered only if there are redundancy, virtualization, management and automation on many levels or layers of infrastructure and applications. From technical perspective (as a hardware oriented view) the management and automation raise the cloud above the known virtualization techniques. From application perspective we have to consider that services can be offered if they exist as separated units and they can be connected directly or in the background. We are also referring to cloud-based solutions in case of application running in a cloud environment. These are not really SaaS examples, but rather application on PaaS or even IaaS bases. The cloud environment what we use can be different:

- **Public clouds:** service providers offer their resources to the public.
- **Private clouds:** designed for exclusive use of a single organization (mainly local).
- **Hybrid clouds:** combining the two previous, part of the infrastructure runs in private, the remaining provides public services.

VPC is another layer over the public clouds, engaging a Virtual Private Network (VPN) protocol, which enables service providers to design and use their own network and security. For more customers it is short transition from datacentre architecture to cloud, while still owning the virtualized network layer [2]. In our examples we show the differences of the solutions offered by larger ERP provides in almost each cases.

Requirements and possibilities in layers

The layered architecture of the cloud offers more control over the responsibility, whereas every layer owner has to focus on the specific services provided their layer. Resource planning is more flexible, because common pools of computing resources can be dynamically assigned to consumers. This is one of the key properties of cloud computing, allowing resource allocation on the fly. When analysing cloud based service, one of the key point is on demand resource allocation. Cloud operation has to satisfy the service level objectives (SLO), with the possible lowest operational cost. The goal is to optimize the mapping of SLOs (like Quality of Service, QoS) to low level resources like CPU and memory usage. Automatized resource distribution for internet based application were studied in the past[3]. The performance model typically based on: a) number of application instances which are required to handle the requirements which satisfies QoS b) predicting future demand and resource needs c) automated resource allocation based on predictions on the future demand. There are several ways to build up these performance model, like statistical machine learning[4] or deep learning algorithm.

This leads to resource usage based pricing, as charging the customer per use bases. Clouds are based on service oriented operating model, so it natural to provide a consistent service level agreement (SLA). Clouds are also frequently based on geologically different located data centres, which can minimize the risk of service outage. Optimally consolidating server in a data centre often described as a variant of the vector bin-packing problem[5] that is an NP-hard problem of optimization. These kinds of activity must affect performance in any way. Shared resource between servers (network, disk storage, etc.) allocation can lead to traffic jam amongst servers, when VM changes its configuration[6].

Possible solutions for virtualization and clouds under ERP services

There are many virtualization theorems and implementations available. [7] The Microsoft environment does not really support the application virtualization (like multi node clustering), though technically the operating system is capable to handle the tasks. In this kind of virtualization the services

always hold their resources (mainly IP addresses and names, storage areas, and user environments). This makes the application (or service) capable to be moved from a host to another one without having configuration problems in the consuming services, application. Main disadvantage of this kind of virtualization is that the service should be stopped before move (like in a HA-cluster solution). The other, nowadays used virtualization model is the kernel level sharing, so-called light way virtualization. It has some history as well because earlier the SUN Microsystems (zone) and IBM (AIX wPAR - Workload Partitioning) implemented already this feature, where isolations are on kernel namespace and control group level. The Linux operating systems offer the LXC (Linux Container) as a solution for that. The Docker solution expanses this feature, by managing the OS container capabilities. It can now manage Windows based containers as well, but the base operating system should be Windows certainly.[8] The last, well-known virtualization is the hypervisor technique. The most popular implementations are the VMware and Hyper-V. One of the main differences between the light way virtualization and hypervisor virtualization is the overhead what the hypervisor layer gives to the environment.

ERP systems as on premise solutions can exploit almost each virtualization techniques, which is good for using basic private cloud environments. For SaaS usage these szstems should have been refactored for reusable pieces as services.

ERP solutions in practice

In our paper we considered two large ERP vendors: Microsoft and SAP. Both vendors have their own on premise solutions, which can run in cloud environment using the IaaS or PaaS types. SAP bought some cloud-based solutions in the last years offering real cloud services (SaaS): Hybris, SuccessFactor or FieldGlass. Around the ERP II. era the Service Oriented Architecture (SOA) was the goal for each vendor or at least to offer a possible enabler for making service orchestration, having a service repository and enabling high level workflows above the services. This was one of the main drivers to the service offering on cloud. The ERP vendors tried to separate and degrade the modules and applications to smaller pieces, which could be simple services. Unfortunately this direction did not bring the expectations. The cloud readiness and the mobility forced the companies to create real service offerings.[9] Microsoft has reconstructed the Dynamics as described later. SAP started this reconstruction, but it is still not fully completed, but some features are available already.

In the practice many companies are still afraid of using cloud technologies, especially public cloud solutions. The thinking, internal functioning is not yet ready for that. Many regulations should also be considered regarding the data sensitivity and location. These companies are thinking about building an own private cloud or using hybrid cloud as a starting point. The public cloud providers have a huge advantage by having the working management and automation toolset for IaaS and PaaS. Some cloud partners offer parallel PaaS and SaaS possibilities by giving add-ons for installing, deploying ERP components on the public cloud. If a company would like to build their own private cloud, it should employ some good experts, or buy consultancy. For long-term investment it can be a good direction.

As the first step the companies try to build a cloud environment (mainly IaaS) for disaster recovery (DR) purposes. They do not consider exactly the costs of moving the production to the DR site located in the cloud and the cost of coming back to the normal on premise functionality. The other main aspect is that it is better to have everything on the same side as using hybrid solution. If more of the applications are on the cloud, the ERP can be implemented there as well.

Additional aspects

Components based software development processes, based on the OOP methodology, supports code reuse strongly. These technologies, like CORBA, J2EE, COM+, .NET, offers a middleware layer, where reusable components can be built. Change management methodologies are need to be revised as the number of companies going into cloud based service oriented operational models are rapidly growing. The technology changes trigger organizational, business operations and software development side changes also. New predefined SLAs are needed for audit trails and software change management. The security structure is also more complex in an agile cloud environment. Most SaaS cloud implementation are based on existing IaaS cloud implementations, for VM and application hosting.

SaaS ERP services are now available for public and private cloud environments as well.

References

[1] Richard Murch. *The Software Development Lifecycle - A Complete Guide* Amazon Digital Services LLC, ASIN: B007ZCRP1I

[2] Zhang, Q., Cheng, L. & Boutaba, R. J. *Internet Serv Appl* 1: 7. doi:10.1007/s13174-010-0007-6 (2010)

[3] Zhang Q et al. *A regression-based analytic model for dynamic resource provisioning of multi-tier applications* In: Proc ICAC (2007)

[4] Bodik P et al. *Statistical machine learning makes automatic control practical for Internet datacenters* In: Proc HotCloud (2009)

[5] Chekuri C, Khanna S. *On multi-dimensional packing problems* SIAM J Comput 33(4):837-851 (2004)

[6] Padala P, Hou K-Y et al. *Automated control of multiple virtualized resources* In: Proc of EuroSys (2009)

[7] A. Selmeci, T. Orosz, Gy. Györök. *Innovative ERP virtualization* ISBN 978-1-4799-0303-0 In: IEEE 11th International Symposium ib Intelligent Systems and Informatics (SISY), 2013, Subotica Serbia, pp. 69-75.

[8] MS Contributors. *MS Web Document* https://docs.microsoft.com/en-us/dynamics365/business-central/dev-itpro/developer/devenv-running-container-development

[9] A. Selmeci, T. Orosz, Gy. Györök. *Teaching ERP User Interfaces: Adequate Sequences of Topics and Technologies* ISBN: 978-1-4673-8739-2 In: IEEE 14th Jubilee International Symposium on Applied Machine Intelligence and Informatics (SAMI), 2016, Herlany Szlovákia, 7 oldal

Strip Constrained Binary Tomography

Judit Szűcs, Péter Balázs

Abstract: The goal of discrete tomography(DT) [1] is to reconstruct discrete images from their projections (sets of line sums taken along parallel lines in different directions). Binary tomography is a specific area of DT when the image to reconstruct is binary. In our case, 1 stands for the object (black) and 0 stands for the background (white) pixels. Due to physical limitations, the projections can be gathered from only a few directions in practice, therefore the reconstruction problem can be highly underdetermined. During the reconstruction process, prior knowledge is often incorporated into an energy function to reduce the search space of feasible solution. Thus, the reconstruction issue is equivalent to a function minimization problem.

Motivated by nonogram puzzles, we introduce a novel prior to describe the expected texture of the reconstructed image: the number of strips in each row and column. We propose an exact deterministic and a stochastic method to solve the problem. The effectiveness of the two methods are compared on artificial data sets from different image classes.

Keywords: binary tomography; reconstruction; nonogram

Acknowledgements

This research was supported by the project "Integrated program for training new generation of scientists in the fields of computer science", no. EFOP-3.6.3-VEKOP-16-2017-0002. The project has been supported by the European Union and co-funded by the European Social Fund.

References

[1] Herman, G.T., Kuba, A. (eds.): Advances in Discrete Tomography and Its Applications. Birkhäuser, Boston (2007)

Lookahead can help in maximal matching

Kitti Gelle, Szabolcs Iván

Abstract: In this paper we study a problems in the context of fully dynamic graph algorithms that is, when we have to handle updates (insertions and removals of edges), and answer queries regarding the current graph, preferably in a better time bound than running a classical algorithm from scratch each time a query arrives.

We show that a maximal matching can be maintained in an (undirected) graph with a deterministic amortized update cost of $O(\log m)$ (where m is the all-time maximum number of the edges), provided that a lookahead of length m is available, i.e. we can "peek" the next m update operations in advance.

Introduction and notation

In the past two decades, there has been a growing interest in developing a framework of algorithm design for *dynamic graphs*, that is, graphs which are subject to *updates* – in our case, additions and removals of an edge at a time. The aim of a so-called fully dynamic algorithm (here "fully" means that both addition and removal are permitted) is to maintain the result of the algorithm after each and every update of the graph, in a time bound significantly better than recomputing it from scratch each time.

Also in [2], a systematic investigation of dynamic graph problems in the presence of a so-called *lookahead* was initiated: although the stream of update operations can be arbitrarily large and possibly builds up during the computation time, in actual real-time systems it indeed possible to have some form of *lookahead* available. That is, the algorithm is provided with some prefix of the update sequence of some length (for example, in [2] an assembly planning problem is studied in which the algorithm can access the prefix of the sequence of future operations to be handled of length $\Theta(\sqrt{m/n}\log n)$), where m and n are the number of edges and nodes, respectively. Similarly to the results of [2] (where the authors devised dynamic algorithms using lookahead for the problems of strongly connectedness and transitive closure), we will execute the tasks in batches: by looking ahead at $t = O(m)$ future update operations, we treat them as a single batch, preprocess our current graph based on the information we get from the complete batch, then we run all the updates, one at a time, on the appropriately preprocessed graph. This way, we achieve an amortized update cost of $O(\log m)$ for maintaining a maximal matching.

In this paper, a graph G is viewed as a set (or list) of edges, with $\|G\|$ standing for its cardinality. This way notions like $G \cup H$ for two graphs G and H (sharing the common set $V(G) = V(H)$ of vertices) are well-defined.

Maximal matching with lookahead

In this section we present an algorithm that maintains a maximal matching in a dynamic graph G with constant query and $O(\log m)$ update time (note that $O(\log m)$ is also $O(\log n)$ as $m = O(n^2)$), provided that a *lookahead* of length m is available in the sequence of (update and query) operations. This is an improvement over the currently best-known deterministic algorithm [3] that that has an update cost of $O(\sqrt{m})$, without lookahead, and achieves the same amortized update cost as the best-known randomized algorithm [1].

In this problem, a *matching* of a(n undirected) graph G is a subset $M \subseteq G$ of edges having pairwise disjoint sets of endpoints. A matching M is *maximal* if there is no matching $M' \supsetneq M$ of G. Given a matching M, for each vertex v of G let $\text{MATE}(v)$ denote the unique vertex u such that $(u, v) \in M$ if such a vertex exists, otherwise $\text{MATE}(v) = \text{NULL}$.

In the fully dynamic version of the maximal matching problem, the update operations are edge additions $+(u, v)$, edge deletions $-(u, v)$ and the queries have the form $\text{MATE}(u)$.

The following is clear:

Proposition 1. *Suppose G is a graph in which M is a maximal matching. Then a maximal matching in the graph $G + (u, v)$ is $M \cup \{(u, v)\}$, if $\text{MATE}(u) = \text{MATE}(v) = \text{NULL}$, and M otherwise.*

This proposition gives the base algorithm GREEDY for computing a maximal matching in a graph:

```
1 Let M be an empty list of edges;
2 for( (u,v) ∈ G )
3   if( MATE(u) == NULL and MATE(v) == NULL)
4     MATE(u) := v; MATE(v) := u;
5     insert (u,v) to M;
6 return M;
```

Note that if one initializes the MATE array in the above code so that it contains some non-NULL entries, then the result of the algorithm represents a maximal matching within the subgraph of G spanned by the vertices having NULL MATEs initially. Also, with M represented by a linked list, the above algorithm runs in $O(m)$ total time using no lookahead. Hence, by calling this algorithm on each update operation (after inserting or removing the edge in question), we get a dynamic graph algorithm with no lookahead (so it uses a lookahead of at most m operations), a constant query cost (as it stores the MATE array explicitly) and an $O(m)$ update cost. Using this algorithm A_1, we build up a sequence A_k of algorithms, each having a smaller update cost than the previous ones. (In a practical implementation there would be a single algorithm A taking k as a parameter with the graph G and the update sequence, but for proving the time complexity it is more convenient to denote the algorithms in question by A_1, A_2, and so on.)

In our algorithm descriptions the input is the current graph G (which is $\emptyset$ in the first time we start running the program) and a sequence $(q_1, \ldots, q_t)$ of operations. Of course as the sequence can be arbitrarily long, we do not require an explicit representation, only the access of the first m elements (that is, we have a lookahead of length m).

Lemma 3. *Assume A_k is a fully dynamic algorithm for maintaining a maximal matching with an $f(k) \cdot m^{1/k}$ amortized update cost, constant query cost using a lookahead of length m.*

Then there is a universal constant c such that there exists a fully dynamic algorithm A_{k+1} that also maintains a maximal matching with $(f(k) + c(1 + \log m))m^{1/k+1}$ amortized update cost, and a constant query cost using a lookahead of length m.

Now, as A_1 is an algorithm satisfying the conditions of this lemma with $k = 1$ and $f(k) = c_0$ for some constant c_0, it implies that for each $k > 1$ that there is a fully dynamic algorithm that maintains a maximal matching with an amortized update cost of $(c_0 + kc(1 + \log m))m^{1/k} = O(k \log m \cdot m^{1/k})$. Setting $k = \log m$ we get that $A_{\log m}$ has an amortized update cost of $O(\log^2 m \cdot m^{1/\log m}) = O\left(\log^2 m \cdot \left(2^{\log m}\right)^{1/\log m}\right) = O(\log^2 m \cdot 2) = O(\log^2 m)$.

Hence we get:

Theorem 1. *There exists a fully dynamic algorithm for maintaining a maximal matching with an $O(\log^2 m)$ amortized update cost and constant query cost, using a lookahead of length m.*

Now we prove Lemma 3 by defining the algorithm A_{k+1} below.

- The algorithm A_{k+1} works in *phases* and returns a graph G (as an edge set) and a matching M (as an edge list).
- The algorithm accesses the *global* MATE array in which the current maximal matching of the whole graph is stored. (A_{k+1} might get only a subgraph of the whole actual graph as input.)
- In one phase, A_{k+1} either handles a block $\vec{q} = (q_1, \ldots, q_t)$ of $t = m^{\frac{k}{k+1}}$ operations or a single operation.
- Let G and M respectively denote the current graph and matching we have in the beginning of a phase.
- If $\|G\|$ is smaller than our favorite constant 42, then the phase handles only the next operation by explicitly modifying G, afterwards recomputing a maximal matching from scratch, in $O(42)$ (constant) time. That is,
  1 We iterate through all the edges $(u, v) \in M$, and set MATE$[u]$ and MATE$[v]$ to NULL (in effect, we remove the "local part" M of the global matching);
  2 We apply the next update operation on G;
  3 We set $M :=$ GREEDY$(G,$ MATE$)$.

Otherwise the phase handles t operations as follows:

1 Using lookahead (observe that $t < m$) we collect all the edges involved in $\vec{q}$ (either by a $+(u,v)$ or a $-(u,v)$ update operation) into a graph G'.

2 We construct the graph $G'' = G - G'$.

3 We iterate through all the edges $(u,v) \in M$, and set MATE$[u]$:= NULL, MATE$[v]$:= NULL.

4 We run $M :=$ GREEDY$(G'',$ MATE$)$.

5 We call $A_k(G \cap G', (q_1, \ldots, q_t))$. Let G^* and M^* be the graph and matching returned by A_k.

6 We set $G := G'' \cup G^*$ and $M := M \cup M^*$.

We will now show its correctness. That is, we claim that each A_k maintains a maximal matching among those vertices having a NULL MATE when the algorithm is called. This is true for the greedy algorithm A_1. Now assuming A_k satisfies our claim, let us check A_{k+1}. When the graph is small, then the algorithm throws away its locally stored matching M, resetting the MATE array to its original value in the process (in fact, this is the only reason why we store the local matching at each recursion level: the global matching state can be queried by accessing the MATE array alone). Then we handle the update and run GREEDY, which is known to compute a maximal matching on the subgraph of G spanned by the vertices having a NULL mate. So this case is clear.

For the second case, if a block of t operations is handled, then we split the graph into two, namely a difference graph G' and an intersection graph G''. By construction, when handling the block, the edges belonging to G' do not get touched. Hence, at any time point, a maximal matching of G can be computed by starting from a maximal matching of G' and then extending the matching by a maximal matching in the subgraph of G'' not covered by the matching of G'. Thus, if we compute a maximal matching M' in the subgraph of G' spanned by the vertices having a NULL MATE, updating the MATE array accordingly (that is, calling GREEDY on G'), and maintaining a maximal matching M'' over the vertices of G'' having a NULL MATE after that point (which is done by A_k, by the induction hypothesis), we get that at any time $M' \cup M''$ is a maximal matching of G. Hence, the algorithm is correct.

Now we analyse the time complexity of A_{k+1}. When a phase handles t operations, then Step 1 can be executed in $O(t \log t) = O(m \log m)$ time (if we use a self-balancing tree representation for storing our graphs, say an AVL tree). Then in Step 2, we construct the difference of the two sets of size $O(m)$ in $O(m \log m)$ time. Step 3 requires an additional time of $O(m)$,since the matching is of size $O(m)$ and it is stored as a list of edges. For Step 4, as $|G''| \leq \|G\| = m$, also an $O(m)$ time is required, and for Step 5, computing the intersection $G \cap G'$ requires a time of $O(m \log m)$, and A_k, being run on a dynamic graph having at most $t = m^{\frac{k}{k+1}}$ edges during its whole lifecycle of t operations needs $t \cdot f(k) \cdot t^{1/k} = m^{\frac{k}{k+1}} \cdot f(k) \cdot m^{\frac{1}{k+1}} = f(k) \cdot m$ computation steps. Gluing together the graphs and the matchings in Step 6 needs a time of $O(m \log m) + O(m)$. Hence the total cost of Steps 1-6 handling a whole phase is $O(m) + O(m \log m) + O(m) + f(k) \cdot m + O(m \log m) + O(m) = (f(k) + c(1 + \log m))m$ for some universal constant c, and since a phase consists of $m^{\frac{k}{k+1}}$ operations, the amortized cost of a single operation becomes $(f(k) + c(1 + \log m))m^{\frac{1}{k+1}}$ and Lemma 3 is proved.

The careful reader may observe that a major part of the time bound comes from the set operations. If an initialization cost of $O(n^2 \log n)$ is affordable (i.e. if there are $\Omega(n^2 \log n)$ operations in total), then we can do better:

- Each algorithm A_k has an adjacency matrix as well, initialized to an all-zero matrix in the very beginning (this initialization takes the aforementioned $O(n^2 \log n)$ setup cost).
- In Step 1, edges of G' are stored into this matrix (taking still $O(m)$ time).
- Now the graphs $G - G'$ and $G \cap G'$, as lists of edges, can be constructed in $O(m)$ time (since lookup in G' now takes constant time instead of the previous $O(\log m)$).
- Since G'' is represented as an edge list, GREEDY still takes $O(m)$ time.
- After performing Step 5, we have to set the auxiliary matrix to an all-zero matrix by looking ahead once again the very same sequence and setting each accessed edge to 0. This takes $O(m)$ time.
- Also, taking the unions of the graphs and matchings upon returning can be destructive to the original lists, thus it can be done in constant time.

Hence in this case the total cost spent for a phase becomes $O((f(k)+c)m)$ for some universal constant c, yielding an amortized update cost of $O((k+1)\cdot m^{1/k+1})$ for A_k, which boils down to an amortized $O(\log m)$ update cost by choosing $k = \log m$ and we showed:

Theorem 2. *There exists a fully dynamic algorithm for maintaining a maximal matching with an $O(n^2 \log n)$ initialization cost, $O(\log m)$ amortized update cost and constant query cost using a lookahead of length m.*

Acknowledgements

This work was supported by the ÚNKP-17-3 New National Excellence Program of the Ministry of Human Capacities.

References

[1] Surender Baswana, Manoj Gupta, and Sandeep Sen. Fully dynamic maximal matching in $O(\log n)$ update time. *SIAM Journal on Computing*, 44(1):88–113, 2015.

[2] S. Khanna, R. Motwani, and R. H. Wilson. On certificates and lookahead in dynamic graph problems. *Algorithmica*, 21(4):377–394, Aug 1998.

[3] Ofer Neiman and Shay Solomon. Simple deterministic algorithms for fully dynamic maximal matching. *ACM Trans. Algorithms*, 12(1):7:1–7:15, November 2015.

Topology-based Classification Error Calculation based on IndoorGML Document

Krisztián Ilku, Judit Tamás

Abstract: Topology-based classification error calculation method for symbolic indoor positioning is presented based on IndoorGML document. Symbolic indoor positions or Zones are well-defined parts of the building, which can be treated as a classification category. The evaluation of well-known classifiers is based on the classical CRISP approach, which considers each misclassification equally wrong. Our previous experimental results revealed the need to consider the topology in the classification error calculation. A possible solution for this challenge is gravitational force based approach, which calculates the classification error by the size and the layout of the Zones. Testing the criteria against this approach in real-life scenario, real-life environment is required. IndoorGML is a standard for specifying indoor spatial information, and it represents the indoor space as non-overlapping closed objects. These indoor spaces are bounded by physical or fictional boundaries, and the representation of an object is by both geometric shape and bounding box. Thus, IndoorGML standard can be used both for modeling the indoor environment and calculation the classification error for symbolic indoor positioning services. In this paper, the gravitational force based approach is examined in real-life environment of Institute of Information Science building in University of Miskolc defined in IndoorGML Document.

Keywords: IndoorGML, Classification Error, Indoor Localization, Topology

Introduction

Indoor Positioning System is considered as an active research field since the early 1990s, and these systems are detailed in the following surveys [1][2]. Absolute, proximity and symbolic indoor positioning are usually distinguished. Absolute location is the coordinates of an object. Proximity position is an orientation to known reference points. Symbolic position refers to the well-defined part of a building, such as rooms, and these positions can be considered as categories. Thus the symbolic positioning can be converted into a classification [3] problem. Classification is a well-studied part of data mining so numerous well-known classifiers could be applied for indoor positioning.

The evaluation of well-known classifier methods is usually based on the classic CRISP approach [4] which treats every misclassification equally wrong. Our previous experiments [5] shows that a more accurate classification method can predict further the symbolic positions from the original location when it is misclassified, than a less accurate classifier. In conclusion, a different approach is recommended to be used as an alternative of CRISP, which considers the topology of the building [6]. Topology of the building defines the Zones and their sizes and arrangement. The classification error should be proportional to the sizes of the Zones and the layout should have a high impact on the error values. In addition, the classification error should not necessary be symmetric due to the size differences.

A gravitational force-based approach had been presented in our previous work [7]. The approach had been tested in a simulated environment, and based on the results, the gravitational-force based approach is a promising candidate to be used instead of CRISP. However, real-life scenarios are essential for calculating classification error for symbolic indoor positioning tasks. Hence, information about a real-life indoor environment is required.

IndoorGML [8] is a standard for representing indoor spaces in an XML formatted document, where it contains the bounding three dimensional absolute coordinates for each symbolic indoor positions. Based on these documents, the sizes and the arrangement of the Zones can be derived. Thus IndoorGML standard can be used in the calculation of classification error for symbolic indoor positioning. The error calculation method is planned to be integrated into the ILONA System [9].

IndoorGML

IndoorGML [8, 10] is a standard of Open Geospatial Consortium (OGC) for specifying a data model of indoor spatial information. IndoorGML defines XML Schemas and it focuses on indoor navigation purposes. Indoor spaces are non-overlapping closed objects and they are bounded by physical or fic-

tional boundaries. Besides the boundaries, for each space an *Envelope* is defined, which is a bounding box given by two diagonally opposite corner points, that can be used for searching.

Environment

The base environment is selected to be the three-story tall Institute of Information Science Building of the University of Miskolc. The building contains an atrium lobby, 26 offices, 15 laboratories, a lecture hall and storage rooms on nearly $3800m^2$.

The floor plan of the building can be seen in Figure 1, where ■ ✚ ★ symbols represent the ground, the first and the second floors. In the floor plan, the narrow represents the entrance of the building, and the dot is the base of the reference coordinate system. The front wall is curved, and the lobbies in the 1st and 2nd floor are bordered with curved railings, hence the shape of these rooms are not consistently rectangular cuboid or box. The statistics of the created IndoorGML Documents [11] can be seen in Table 1.

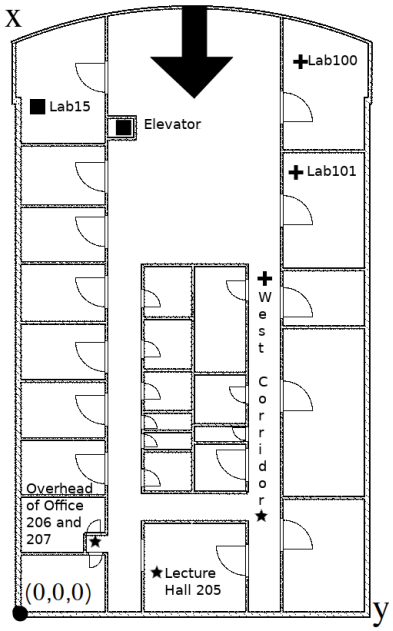


Figure 1: Floor Plan in Institute of Information Science Building

Table 1: Statistic of IndoorGML Documents of Miskolc IIS Building

Floor	Cell	Linear	Curved
Ground Floor	18	17	1
First Floor	33	32	1
Second Floor	21	20	1
Total	72	69	3

Method

The gravitational force based approach [7] should fulfill the requirements for considering the topology. The method assumes the disjunction of the Zones, and requires the determination of capacity (V) and distance (d) functions. The gravitational force [12] measures the similarity between two Zones, and it is proportional to the product of their capacity and inversely proportional to the square of their distance. The formulation of the gravitational force can be seen in Equation 22.

$$F_g(Z_i, Z_j) = \frac{V(Z_i)V(Z_j)}{d(Z_i, Z_j)^2} \tag{22}$$

While the gravitational force represents the similarity between Zones, their difference is required for error calculation. The δ function can be calculated as seen in Equation 23. It represents the difference of two zones in the $[0, 1]$ range.

$$\delta(Z_i, Z_j) = \frac{1}{1 + F_g(Z_i, Z_j)} \tag{23}$$

The classification error should be also proportional to their sizes of the zones and be asymmetric due to these size differences. The ϵ function had been introduced to measure the classification error in the $[0, 1]$ range, as it can be seen in Equation 24.

$$\epsilon(Z_i, Z_j) = \frac{V(Z_i) * \delta(Z_i, Z_j)}{V(Z_i) + V(Z_j)} \tag{24}$$

Results

For the implementation of the topology-based classification error calculation a Java application had been developed. It converts the data from the IndoorGML XML document to Java classes, both provided by IndoorGML. A zone is represented as CellSpace object, which means that each zone contains the name and the ID of the zone, the bounding and two diagonal cornerstone coordinates.

In this paper, the three-dimensional lower- and upper corner coordinates are used in distance and capacity calculation. For distance calculation the Euclidean distance had been chosen, and the distance is specified by the length of the straight line between the middle points of the two zones. To calculate the capacity of a zone, the benefit of cuboid property had been applied to calculate the volume. Based on the distance and the capacity function, the classification error can be calculated for each zone pair. Table 2 shows some examples of the classification error.

Table 2: Examples of the Topology-Based Classification Error

Actual			Predicted			Error
Name	Capacity (m^3)	Centroid	Name	Capacity (m^3)	Centroid	
Ground Floor Elevator	17.5	(40.8,8.3,4.5)	Overhead of Office 206 and 207	2.8	(5.5,6.5,4.5)	0.3641
Lab15	169.4	(45.5,2.8,1.4)	Lecture Hall 205	156.8	(3.5,14,7.6)	0.0009
Lab100	241.5	(44,8,24.8,4.5)	Lab101	172.9	(33.8,25.3,4.5)	0.0002
2nd Floor West Corridor	231	(16.5,19.8,7.6)	1st Floor West Corridor	231	(16.5,19.8,4.5)	0.0001

The overall average classification error is 0.0088 with 0.0271 standard deviation. The highest error calculated is 0.3641 in the case of *Ground Floor Elevator* actual zone is misclassified as *Overhead of Office 206 and 207*. The volume of *Ground Floor Elevator* is $17.5\ m^3$, while the *Overhead of Office 206 and 207* is $2.8\ m^3$.

The two farthest zones are *Lab15* on the front of the ground floor and *Lecture Hall 205* on the back of the second floor. The volume of these zones are 169.4 and $156.8\ m^3$. The classification error calculated in this case is 0.0009.

The *Lab100* and *Lab101* are neighbouring zones with 241.5 and $172.9\ m^3$ volume. The classification error in this case is 0.0002. The *2nd Floor West Corridor* and the *1st Floor West Corridor* zones are congruent, they only differ in the z coordinate. The misclassification in both direction result the 0.0001 value.

Discussion

The gravitational force-based method should consider the topology of the classification error calculation. It is proportional to the sizes and the arrangement of the Zones.

As it can be seen in Figure 1, the *Ground Floor Elevator* and the *Overhead of Office 206 and 207* both relatively small zones, and they are very far from each other. Hence the classification error in this case is high. Otherwise, the *Lab15* and the *Lecture Hall 205* are also very far from each other, but they are both relatively large, thus it has significantly lower error value than in the case of *Ground Floor Elevator* and

Overhead of Office 206 and 207. Therefore the method considers the size and the layout of the zones, thus the misclassification of smallest, farthest zones results the highest error values.

The misclassification of *Lab100* to *Lab101* zones has a relatively small error value. *Lab100* is larger than *Lab101*, but both are considered as relatively large zones and they are neighbouring.

The *2nd Floor West Corridor* and *1st Floor West Corridor* zones are congruent in size and location besides the z coordinate. As it would be expected, the classification error of these two zones are symmetric, and small.

The results of the classification error calculation show that the gravitational force-based approach considers the topology in classification error calculation. However, the calculated error values has a very low average, and the standard deviation should be higher, and the highest error value is lower than the half of the possible range.

Summary

To demonstrate the gravitational force-based classification error calculation approach, a Java application was developed as a proof of concept. The selected environment is the Institute of Information Science building in University of Miskolc. A model of this building had been constructed with IndoorGML standard. The IndoorGML Documents specify indoor spatial information, and they represents the indoor space as non-overlapping cells. The presented results showed that the gravitational force-based approach fulfills the criteria to consider the topology in classification error calculations. Thus it is proportional to the sizes and the layout of the zones, and it is not necessary symmetric. Therefore, the gravitational force-based approach can be an alternative to the CRISP approach in the evaluation of symbolic indoor positioning methods. Hence, the developed Java application can be used to calculate the classification error based on IndoorGML Documents.

In the future, the gravitational force-based approach will be integrated to the ILONA System. It will be also used to compare well-known classifiers for selecting the most suitable for symbolic indoor positioning purposes.

Acknowledgements

The described article was carried out as part of the EFOP-3.6.1-16-2016-00011 "Younger and Renewing University – Innovative Knowledge City – institutional development of the University of Miskolc aiming at intelligent specialisation" project implemented in the framework of the Szechenyi 2020 program. The realization of this project is supported by the European Union, co-financed by the European Social Fund.

References

[1] Hakan Koyuncu and Shuang Hua Yang. A survey of indoor positioning and object locating systems. *IJCSNS International Journal of Computer Science and Network Security*, 10(5):121–128, 2010.

[2] Hui Liu, Houshang Darabi, Pat Banerjee, and Jing Liu. Survey of wireless indoor positioning techniques and systems. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 37(6):1067–1080, 2007.

[3] Krzystof Jajuga, Andrzej Sokolowski, and Hans-Hermann Bock. *Classification, clustering, and data analysis: recent advances and applications*. Springer Science & Business Media, 2012.

[4] David Martin Powers. Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. 2011.

[5] Judit Tamas and Zsolt Toth. Classification-based symbolic indoor positioning over the miskolc iis data-set. *Journal of Location Based Services*, 12(1):2–18, 2018.

[6] Judit Tamas and Zsolt Toth. Limitation of crisp accuracy for evaluation of room-level indoor positioning methods. In *2018 IEEE International Conference on Future IoT Technologies*, Eger, Hungary, 01.18-01.19 2018. IEEE. In press.

[7] Judit Tamas and Zsolt Toth. Topology-based classification error calculation for symbolic indoor positioning. In *Carpathian Control Conference (ICCC), 2018 19th International*, pages 643–648. IEEE, 2018.

[8] J Lee, Ki-J Li, S Zlatanova, TH Kolbe, C Nagel, and T Becker. Ogc® indoorgml. *Open Geospatial Consortium standard*, 2014.

[9] Zsolt Toth. Ilona: indoor localization and navigation system. *Journal of Location Based Services*, 10(4):285–302, 2016.

[10] Ogc indoorgml-with corrigendum. http://docs.opengeospatial.org/is/14-005r4/14-005r4.html. [Online; accessed 13-October-2017].

[11] Krisztian Ilku and Judit Tamas. Indoorgml modeling: A case study. In *Carpathian Control Conference (ICCC), 2018 19th International*, pages 633–638. IEEE, 2018.

[12] Isaac Newton. *The Principia: mathematical principles of natural philosophy*. Univ of California Press, 1999.

Industrial process modelling with operations research method

László Péter Pusztai, Balázs Kocsis, István Budai, Lajos Nagy

Abstract: The operation of a business process is not as easy as it seems for the first time. A lot of complex connections can be identified in a production process, while the performance of a machine can be either uncertain or unknown. These factors could result in inappropriate conclusions and decisions. Collecting data is an essential part of business process modelling, while operations research methods can provide a good evaluation tool for making the right decisions. The article aims at presenting a process rationalization while total process time optimization is carried out based on a minimal extra cost investment. The examined variables of the optimization were total process time and cost of human resource. The target value of total process time reduction was 10%.

Keywords: business process, network model, total process time reduction, operations research

Introduction

The aim of every company is to earn higher profit. This can be accomplished by many approaches. However, one of the most important way to achieve the optimal process time is by making process improvements. An extended network model can help to cut down the value of this indicator by selecting and scheduling the added resources. This technique can only be applied when it is possibile to reduce the total process time by investing in extra employees or machines, and when the process can perform less output than demanded by the customer. This article is organized as follows. Section 2 gives a brief introduction about operations research and network models. Section 3 presents an industrial process as a case study and the applied research methodology. Section 4 shows the original model as well as the extended network model. Finally, in the last chapter, conclusions are provided.

Theoretical Background

Operations research

The operations research is a methodology, which was invented in the middle of the 20th century when optimal solutions to military problems were searched for by applying mathematics [1]. After World War II., these mathematical models were shared and widely-used in both scientific and practical fields, such as production scheduling[2], agriculture[3], and this list could be expanded. This methodology aims at achieving the optimal solution (optimization) in the most efficient way[4]. During the calculation of an optimization, input combinations are looked for, which meet the analysts' goals the most. From the mathematicians' point of view, the most challenging task is to represent real life by mathematical models, because business processes usually have more than one variables, for example a lot of inputs and outputs[5].

A mathematical model is based on an objective, which is an n-variate *f* function, whose value must be optimized:

$$f(x_1, x_2, ...x_n).$$

There are constraints in the model, which provide a limit for the possible set of solutions. And finally, there is the set of possible solution, which is indicated by S, and this is included in Rn set [4]. In order to find an optimal solution for a production problem, network models must be applied [6].

Network models

Networks are known in many fields of use, such as routes, railways, communications, drinking and sewage networks, electrical circuits [1]. A common feature in them is that the direction of the delivery is strictly regulated. Another common characteristic is that the flow capacity of the routes in a given network is limited [4]. A general network model can the demonstrated with a G(N,A) directed graph, which consists of the following basic elements: nodes that deliver items (information, water, etc.) to the next node(s). The arcs represent logical connections between nodes, while capacity determines the limits

of flow. With the application of network modelling, project management tasks can be easily solved. Especially, Critical Path Method (hereinafter CPM) and Program Evaluation and Review Technique (PERT) can be easily executed regardless of the size and complexity of the models. CPM is mostly used in the planning phase of a project production. Furthermore, this method can be useful for modeling manufacturing processes, because both the total process time and the critical path with the biggest impact on the production throughput can be determined. In addition, time slacks in the production can be analyzed, which are used for production scheduling in a very efficient way.

Methodology

Process presentation

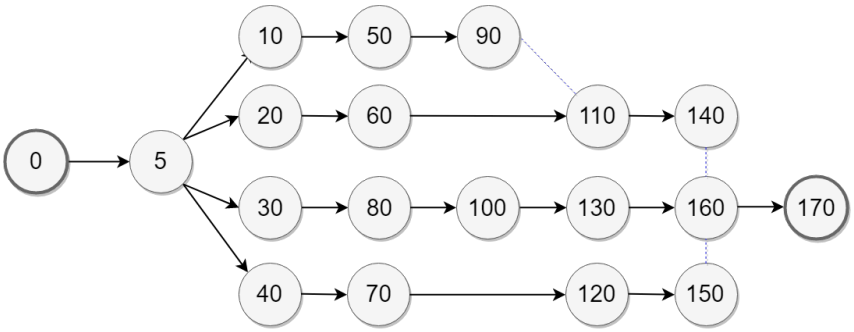
The manufacturing process produces the final product from 4 raw materials, later on indicated as Material A, B, C and D. The production process is made up of 4 different transformation process activities, plus a pre-assembly as well as an assembly stage, as can be seen in *Table 1*.

Table 1: Nodes belonging to workstages

Stages	Stage 1	Stage 2	Stage 3	Stage 4	Pre-assembly	Assembly
Nodes	10, 20, 30, 40	50, 60, 70, 80	90, 100	110, 120, 130	140, 150, 160	170

Two of the above mentioned raw materials are processed in every process stage, while the other two undergo only three transformation acitivites in their production before they reach the assembly stage. Two operators work in each workstage. The graph of the production process can be seen in *Figure 1.:*

Figure 1: Graph of the manufacturing process



In order to accomplish the process, dummy activities must be added to the production. The costs of these activities are 0. The manufacturing company produces in high batches, which means 100 pieces of product per batch. In the following parts of this article, total process time and other values are calculated according to this batch size.

Basic data

Measurements were made to determine the average activity times. The results of the normality tests carried out on each activity had a normal distribution. For the company's request, activity times were distorted and presented in time measurement unit (TMU), which are shown in *Table 2.*:

Table 2: Activity times (in tmu)

Acvitity no.	Avg. time	Acvitity no.	Avg. time	Acvitity no.	Avg. time
0-5	10	40-70	8	100-130	216
5-10	240	30-80	136	110-140	156
5-20	27	50-90	240	120-150	162
5-30	119	80-100	64	130-160	400
5-40	16	90-110 ()	0	140-160 ()	0
10-50	130	60-110	262	150-160 ()	0
20-60	30	70-120	130	160-170	146

Results

Original network model

Based on the information in the above table, the following equations were calculated in order to get the total process time of the production and to identify the critical activities:

$$-x_0 + x_5 \geq 10$$

$$\vdots$$

$$-x_{160} + x_{170} \geq 146$$

Objective:

$$-x_0 + x_{170} \rightarrow MIN!$$

Linear programming was applied to the network model. According to the results, the total process time of the manufacturing process is 1091 TMU. By evaluating the sensitivity analysis report of the model, these activities (nodes) turned out to be critical in the determination of total process time:

$$0 - 5 - 30 - 80 - 100 - 130 - 160 - 170.$$

Extended network model

The aim of the improvement was to reduce the total process time by 10%, which was equal to 982 TMU at this company. This reduction could be accomplished only by hiring new operators, because the company's financial status did not allow for investments in new machines. According to the technology and safety regulations of the production, maximum 3 additional workers can be employed at each stage. This kind of extra work could result in a 5-15% decrease in activity time depending on the workstage. In order to get realistic results, the network model must be changed and extended:

$$-x_0 + x_5 + x_{0-5} \geq 10$$

$$-x_5 + x_{10} + x_{5-10} \geq 10$$

$$\vdots$$

$$-x_0 + x_{170} \geq 982$$

Objective:

$$Cost \rightarrow MIN!$$

In this model, the total process time was a constraint, while the objective of the model was to achieve the desired total process time at the lowest possible cost.

Activity times cannot be reduced to zero by increasing the number of operators. Every workstation requires different operator skills, therefore, the cost of human resources employment differs for each activity. In the last row of *Table 3*, the amounts of time reduction per operator are indicated:

Table 3: Employee data

Stages	Stage 1	Stage 2	Stage 3	Stage 4	Pre-assembly	Assembly
Max reduction (TMU)	160	141	96	86	148	48
Costs (MMU)	125	210	150	240	130	300
Help/operator (TMU)	30	20	25	30	10	15

According to the results of the extended network model, total process time reduction can be carried out at a cost of 13 725 money measurement unit (MMU), which is the lowest possible cost taking constraints into consideration. The model's solution row provided the following recommendations for process improvement: 89 tmu extra work at the 1st process stage and 20 tmu extra work at the pre-assembly stage are necessary. 10% (109 tmu) decrease in total process time can be carried out by hiring three new employees in the 1st stage, and two operators in the pre-assembly stage. It means 5 new employees in total that equals 137.25 unit cost increase in the production. With the solution of the model, the critical path of the process did not change.

Conclusion

This article presents a case study on the optimization of a manufacturing process. If there is a possibility for reducing total process time, the cost of the reduction activities becomes an important factor, because expenses on improvement must be optimized to achieve the desired indicator at the lowest cost. In the presented case study, suggestions were proposed to achieve the 10% total process time decrease at a manufacturing company.

Acknowledgements

This research was supported by the NTP-NFTÖ-17-B-0509 project. The project was financed by the Hungarian Ministry of Human Capacities and the Human Capacities Grant Management Office.

References

[1] T. József - V. Zoltán. Operációkutatás, Akadémiai Kiadó, Budapest, 2014.

[2] Fantahun M. Defersha. Linear programming assisted genetic algorithm for flexible job-shop scheduling with lot streaming, *Computers & Industrial Engineering*, 2018. Vol. 117. pp. 319-335.

[3] Cs. Margit, G. Tímea. Optimization of the production structure of field energy crops, *Annals of the University of ORdaea Economic Science 1*, 2016. pp. 527-537.

[4] V. Béla. Operációkutatási modellek, Typotex Kiadó, Budapest, 2009.

[5] Wayne L. Winston. Operations Research: Applications and Algorithms. Duxbury Press. 2003.

Preliminary Concepts for Requirements Mining and Classification using Hidden Markov Model

László Tóth

Abstract: Requirements specifications are crucial documents of the software development. These documents are created based on the expectations of stakeholders and the requirements of various regulations. The expectations of stakeholders might contain some ambiguities, inconsistencies and also some contradictions especially comparing with regulations. Creating specifications which do not contain issues mentioned above is one of the most important tasks of the business analysts. Reconciling the expectations and requirements can be a demanding task particularly in case of a complex system. Several attempts have been made to support the duties of business analysts using computer-aided natural language processing methods for requirements engineering. One of the most important steps during the elicitation process is the classification of the requirements collected from stakeholders considering that these requirements will be part of the formal and specific models. Investigations devoted to this task have achieved remarkable results using supervised and semi-supervised methods. However, these models need somehow prepared requirements in order to use them as their inputs. The approach presented in this article focuses on extracting and classifying requirements from unstructured documents. Hidden Markov Models are utilized in various field of natural language processing and their usability already has been proven. The idea of using HMM for processing requirements is stemmed from the success of using it for those tasks where extracting information often hidden in unstructured texts is a crucial part of the particular task. Using HMMs, which is a novel approach to processing texts containing requirements, can help also utilize various linguistic features of the sentences that could be obtained with difficulty by classical processes.

Keywords: HMM, HHMM, AHMM, NLP, Requirements, Ontologies

Introduction

Requirements engineering is a vital part of the software development process. This task is accomplished by business analysts based on regulations of the specific business area and interviews which are come from stakeholders. The quality of the resulting specification has a significant impact on the software quality and also the effort needed to develop the software product. Specifications written poorly have heavy consequences for the quality, the budget and also can make a hinder for the maintenance [Firesmith, 2007].

The basis of the requirements specification is made up documents and memos written in natural languages, therefore natural language processing methods can play an important part for supporting business analysts in elicitation processes. Several investigations with considerable results have been made in order to extract and classify requirements from these documents such as [Rashwan et al., 2013], [Casamayor A, 2010], [Cleland-Huang et al., 2007], [Robeer et al., 2016] and [Abad et al., 2017]. Researchers have been focused on the categorization of requirements using supervised and semi-supervised methods. In order to support the finding of the specific texts in unstructured documents, some researchers have applied ontology-based classification methods focusing on the recognition of non-functional requirements [Al Balushi et al., 2007], [Rashwan et al., 2013]. Rashwan et al. [Rashwan et al., 2013] have also created an annotated corpus which is based on the ISO/IEC 9126-1:2001 standard to support classification processes for requirements when there are only a few labeled examples available. [1]

Ontology-based methods might have an important role in the further researches to find an adequate method for minimizing the manual processes regarding requirement elicitation. Ontologies make an important part also in our idea which is based on utilizing the power of Hidden Markov Models in extracting and classifying requirements originated from various textual inputs. There has been no known research focusing on utilizing Hidden Markov Model for requirements engineering tasks, therefore our research as using it for this purpose is a novel approach for harnessing the power of the model.

[1] This standard has been revised and a new standard was published in 2011 which is the ISO/IEC 25010:2011.

Concepts

As mentioned in the previous section several experiments have been accomplished in order to utilize various a priori information during the process of requirement elicitation. Rashwan et al. have built up an ontology-based corpus which content based on the ISO 9126 standard. The content of the corpus was annotated manually and those examples which annotation was agreed among the participants have been retained in the database called by authors as gold standard [Rashwan et al., 2013]. Databases like gold standard mentioned previously can support elicitation process in different business environments without using manual labeling.

In regards to lack of the labeled examples semi-supervised learning approach can play a crucial role in requirements classification and also their identification. Despite the supervised learning methods have decreased the manual work significantly, tools based on these processes have not spread yet. Semi-supervised learning processes and tools are extremely useful where there are only a few labeled examples available. Casamayor et al. have applied Expectation Maximization strategy along with Naive Bayes method for classification of non-functional requirements and their result has surpassed the accuracy and the precision of supervised methods despite the reduced number of labeled examples [Casamayor A, 2010]. Their experiments have shown that increasing the amount of the labeled examples the accuracy and the precision of the model is increasing as well. Semi-supervised methods by their performance and usability can be good candidates for tools supporting the elicitation process however their results have to be checked manually.

Combining semi-supervised methods and ontology-based approach the increase of performance and usability is expected. Furthermore considering that requirements are given as a textual information which is a sequence of linguistic elements and processing it using methods designed for sequence processing we might extract information which is not obtainable for classical methods. This approach provides some other possibilities such as using n-gram samples or exploiting the relationship among the parts of speech or considering also the modality of the given sentence. Hidden Markov Models are often used in various field of sequence processing such as speech recognition, part of speech tagging, named-entity recognition, text summarization, text classification or topic segmentation [Gao and Zhu, 2013], [Al-Anzi, 2017]. The aim of our research is to examine the possibility of using Hidden Markov Models and ontology-based databases for requirements extraction and classification processes.

HMM for Requirements Classification

Requirements classification can be considered as one of the areas of utilizing NLP techniques for requirement engineering. Using HMM for requirement engineering is a novel approach, where the classification is that specific field where this attempt is to be applied first for the sake of comparison with results achieved by using classical methods.

Hidden Markov Models are statistical models where the system which is modeled by HMM can be described as a sequence of observed and unobserved states. In these models, the unobserved states of the given sequence have a property called Markov property which means that the given state of the process only depends on the previous state. This property can be written formally as:

$$P(X_m = x_m | X_{m-1} = x_{m-1}, ..., X_1 = x_1) = P(X_m = x_m | X_{m-1} = x_{m-1})$$

Observable sequences are the series of words which is the sentence to be classified. Defining hidden states for our purpose is considered as a research question. The simplest answer is that we can use classes of requirements for this purpose. We can construct binary classifiers for each class where hidden states can be reduced to only two states. We can also utilize the hierarchical hidden Markov models (HHMM) in order to process the results given by binary classifiers and making the final decision. We can construct models with much more hidden states in order to exploit other linguistic features of the given sentence such as modality or the relationship among the parts of speech.

In the Markov models, two kinds of probabilities are defined. The first one is the transition probability which is conceived between two hidden states, and the emission probability which is the probability of the observed element given in a specific state. The emission probability can be calculated based on ontologies or labeled examples. What other information can be used from ontologies in order to construct our model is also an open question.

Another interesting approach to classify requirements is using Aspect Hidden Markov Models (AHMM) which was proposed by David M. Blei and Pedro J. Monero for topic segmentation tasks in their *Topic Segmentation with an Aspect Hidden Markov Model* article [Blei and Moreno, 2001]. Their work is based on Hoffman's aspect model [Hofmann, 1999]. The mentioned model can be described as a group of probability distributions over two discrete random variables. These random variables are the labels of the specific requirements and the words of the given sentence in our case. The supposition of the model is that the sentence and words are independent of each other, given a specific label. Then the joint probability can be written as [Blei and Moreno, 2001]:

$$P(d, w, z) = P(d|z)P(w|z)P(z)$$

where d denotes the document or the sentence in our case, w denotes the words and z denotes the labels. Although the given supposition is strong this model has performed well in the topic segmentation tasks [Blei and Moreno, 2001].

The various HMMs demand that their inputs be preprocessed according to the attributes of the particular model and the information to be extracted. To find appropriate methods for preprocessing texts containing requirements without losing important information is another research question. Procedures applicable for topic segmentation might be a good starting point.

Due to the lack of relevant researches corresponding to apply Hidden Markov Models for requirements engineering tasks defining a model for requirements classification is an open topic. HMMs can be used for supervised learning if there are a lot of labeled examples available which is unfortunately not the case. Using ontologies, semi-supervised learning can be a reasonable choice. We can construct binary classifiers for each class and these classifiers can be used for a hierarchical model which is applicable for making the final decision. We can also consider those cases where sentences can be classified into more than one classes.

Conclusion

Using Hidden Markov Model for requirements classification is a novel approach to this problem. There are plenty of open questions emerging at the beginning of the research. The first question is the architecture of the model. We have presented some ideas in this article regarding this question however a lot of other constructions can be taken into. One of our future goals is to find an efficient architecture which can support both the extraction and the classification processes.

The second problem is the stability of the model. Computed probabilities regarding using sparse matrices can be very small therefore smoothing techniques are to be applied. Choosing the appropriate smoothing method which fits best for our purpose is another important research question.

We propose further possibilities to improve usability and performance of the extraction and the classification processes such as exploiting relationships among the part of speech, using n-gram models and harnessing the information provided by ontologies. In the long term, the main purpose of our research is developing a tool which can be used to support the duty of business analysts in requirements elicitation processes.

References

[Abad et al., 2017] Abad, Z. S. H., Karras, O., Ghazi, P., Glinz, M., Ruhe, G., and Schneider, K. (2017). What Works Better? A Study of Classifying Requirements. In *Proceedings - 2017 IEEE 25th International Requirements Engineering Conference, RE 2017*, pages 496–501.

[Al-Anzi, 2017] Al-Anzi, F. S. (2017). STATISTICAL MARKOVIAN DATA MODELING FOR NATURAL LANGUAGE PROCESSING. *International Journal of Data Mining & Knowledge Management Process (IJDKP)*, 7(1).

[Al Balushi et al., 2007] Al Balushi, T. H., Sampaio, P. R. F., Dabhi, D., and Loucopoulos, P. (2007). ElicitO: A Quality Ontology-Guided NFR Elicitation Tool. In *Requirements Engineering: Foundation for Software Quality*, pages 306–319. Springer Berlin Heidelberg.

[Blei and Moreno, 2001] Blei, D. M. and Moreno, P. J. (2001). Topic segmentation with an aspect hidden Markov model. *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval SIGIR 01*, 12(2-3):343–348.

[Casamayor A, 2010] Casamayor A, Godoy D, C. M. (2010). Identification of non-functional requirements in textual specifications: A semi-supervised learning approach. *Information and Software Technology*, 52(4):436–445.

[Cleland-Huang et al., 2007] Cleland-Huang, J., Settimi, R., Zou, X., and Solc, P. (2007). Automated classification of non-functional requirements. *Requirements Engineering*, 12(2):103–120.

[Firesmith, 2007] Firesmith, D. (2007). Common requirements problems, their negative consequences, and the industry best practices to help solve them. *Journal of Object Technology*, 6(1):17–33.

[Gao and Zhu, 2013] Gao, X. and Zhu, N. (2013). Hidden Markov model and its application in natural language processing. *Information Technology Journal*, 12(17):4256–4261.

[Hofmann, 1999] Hofmann, T. (1999). Probabilistic Latent Semantic Analysis. *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence (UAI'99)*, 50:2.

[Rashwan et al., 2013] Rashwan, A., Ormandjieva, O., and Witte, R. (2013). Ontology-Based Classification of Non-functional Requirements in Software Specifications: A New Corpus and SVM-Based Classifier. In *2013 IEEE 37th Annual Computer Software and Applications Conference*, pages 381–386. IEEE.

[Robeer et al., 2016] Robeer, M., Lucassen, G., van der Werf, J. M. E. M., Dalpiaz, F., and Brinkkemper, S. (2016). Automated Extraction of Conceptual Models from User Stories via NLP. In *2016 IEEE 24th International Requirements Engineering Conference (RE)*, pages 196–205. IEEE.

Monocular Estimation of 3D Poses from a Distance

Márton Véges, Viktor Varga

Abstract: Most 3D pose estimators only estimate egocentric coordinates where the body is centred at the origin. This is suitable for scenes with a single person but for images with interacting persons it is insufficient. We propose a monocular depth estimator for telephoto lenses to estimate 3D coordinates centred at the camera.

Our method fuses a depth map predictor and a relative 3D pose estimator by means of a 3-layer neural network. We compare the algorithm with the state-of-the-art method and show a 19% improvement.

Introduction

In activity recognition one of the important input features is the human skeleton positions in 3D or 2D space [3][12]. There are many methods to predict 3D positions from a single input image [6][2]. These methods usually only predict the relative coordinates of the body joints in an egocentric coordinate system. In remote situations involving multiple people, the global coordinates of the individuals are needed for estimation.

To this end, we use a monocular depth estimator that infers the distance of each pixel from the camera from a single image. We note that while this is an inherently ambiguous problem, there are multiple features in a scene that can help to infer the real coordinates of objects. Examples are tables, shadows or people whose size vary on a relatively small scale.

The task is difficult since (i) we are lacking depth information, (ii) there is a rich variety of body configurations, (iii) self-occlusion is frequent in monocular viewing, and (iv) small errors around the boundaries of the limbs can cause a joint placed in the background. These limitations can result in large prediction errors.

To mitigate the above problems we combine a depth estimator with a relative 3D pose estimator using a fusion network. We test the algorithm on the NTURGB-D database containing multiple people and find that the mean absolute error is below the half of the average distance between two persons over a variety of actions. These actions include hugging, object passing and handshakes among others.

We are aware of a single paper that targets the same problem [7]. Our method performs better on NTURGB-D by a large margin.

Related Work

Depth estimation Recent models use variations of convolutional neural networks. Laine et al. trains a fully convolutional model with residual up-projection blocks [5]. In [13] a fully convolutional network is trained using synthesised data from video sequences. The training process is unsupervised, all supervisory data is coming from synthetically generated frames.

3D pose estimation 3D pose estimators fall into two groups: methods in the first one predict 2D positions from images first and then predict the final 3D coordinates. Methods in the second one do the task in one step without estimating intermediate 2D positions. In the first category Martinez et al. [6] uses the Stacked Hourglass network [9] for 2D pose estimation and then applies a 5 layer fully connected network with residual connections to regress 3D coordinates. In the second category, Dabral et al. [2] train a network end-to-end applying structural losses on bone lengths and angles.

Global pose estimation Estimating global coordinates and not relative ones is a much less common objective in the literature. In [7] the authors apply a least squares minimisation between the predicted 3D and 2D coordinates. Vnect extends this to videos with additional loss components ensuring temporal smoothness [8].

Method

The proposed pipeline has four stages. First, we run the state-of-the-art OpenPose joint detector [1]. The result of the algorithm is the locations in pixels of 18 joints (in total $18 \cdot 2 = 36$ values) for each of the bodies in the image.

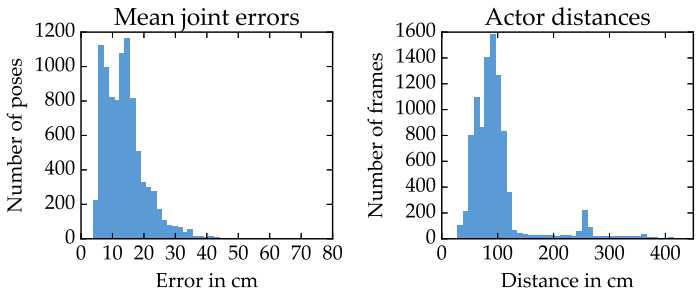


Figure 1: Left figure: distribution of the mean per joint error [4] of our model over all body poses. Right figure: distribution of the distances between the hips of actors on frames with two persons. All distances are in centimetres.

Second, we take the estimated 2D image coordinates of the joints and use the method of Martinez et al. [6] to convert them to relative 3D coordinates. At this stage, the hip is centred at the origin.

Third, a depth predictor trained on the NYU Depth V2 dataset [11] is run (referred as FCRN-D later on) [5]. It returns an estimation of the depth for each pixel in the input image. Note that the scale is ambiguous due to the difference between the calibration of the cameras that took the input image and the images in the training set.

Finally, we employ a fusion network to predict the final depth of the hip. The network has three dense layers with a residual block and batch normalisation applied for additional regularisation. The input is the image coordinates of the 18 joints generated in the first step and the corresponding depth values from the predicted depth map. The output is the distance of the hip from the camera. We get the final coordinates by calculating the coordinates of the hip in a camera centred coordinate system and moving the skeleton there.

Experiments

Database For the experiments the NTURGB-D database [10] was used. It contains 60 different actions performed by 8 actors taken in an indoor setting from 3 different camera angles. Since we are interested in multi-person interactions, we selected only the video sequences containing two persons. The data was split in training and test set by actors: videos of actors #1 and #2 formed the test data. Only camera setup #1 was used. In total the training set contained 370 videos, 25502 frames and 50404 body poses while the test contained 124 videos, 9890 frames and 19345 body poses.

The annotations of the database are based on Kinect that results in a couple of incorrect labels. To check whether these have any effect on the performance, we also ran the algorithm on a subset of the full database: only those body poses are kept where the ground truth joints were at most 40 pixels away from OpenPose's detection. 40 pixels corresponds roughly to 10 centimetres in real coordinates. Since OpenPose's accuracy is very high, it is a strong indicator whether an annotation was correct or not. This subset of the database is referred as Filtered DB later.

Error metric To evaluate the results we used the standard mean per joint position error which is the average Euclidean error over all joints and all poses [4].

Algorithms We compare our method to that of Mehta et al [7]. They minimise the projection error between the predicted 2D coordinates and the translated relative 3D coordinates. We also include a variation of our pipeline to show that the final stage fusion network is indeed needed. In this variation, the depth of the hip is estimated directly from the FCRN-D depth map. To overcome the issue of different calibrations, a simple linear regression is used between the predicted depths and the real distances from camera.

Table 1: Mean joint errors on the NTURGB+D database in centimetres. Best result selected in bold.

	Whole DB	Filtered DB
w/o Fusion	43.0	-
Baseline [7]	17.1	16.4
Ours	**13.8**	**12.9**

Discussion

The results of our experiments are summarised in Table 1. Our method improves 3.3 centimetres over the baseline which is a 19% decrease in the error, reaching state-of-the-art results. Since the average error hides a lot of details, we also present the histogram of the errors over all body poses in Figure 1 (left graph). The right graph in Figure 1 shows the distribution of the distances between the hips of the two actors on the images. The 5th percentile of interpersonal distances is 50.2cm while the median is 88.4cm. The median and 95th percentile of average errors are 13.1cm and 26cm respectively. Hence, we can tell the difference between the people in the images in around 95% of the cases.

We have also run ablation studies to analyse the components of our system. First, when using only the depth estimation of FCRN-D without our fusion network, we get an error of 43cm. This shows that a significant improvement is coming from the network and it is not only FCRN-D producing the results. Note, while the error of 43cm is far below the baseline it is still less than half of the median interpersonal distance.

Second, since the database contains erroneous annotations, we tested whether the improvements are only due to gains on the outliers or not. For this we used a subset of the full test set keeping only frames where the annotations were close to OpenPose's detections in 2D. The results show that our model is still better than the baseline (12.9 vs 16.4cm) indicating that learning mislabellings are not the root cause of the better performance.

Conclusions and Future Work

We have showed that our pipeline produces state-of-the-art global pose estimation on the NTURGB-D database. The lower error is not a result of the off-the-shelf components or incorrect annotations.

Our method can be extended in several ways, e.g. handling outdoor scenes or occlusion of body parts by objects. Also, end-to-end learning is a popular approach where the components are not treated separately but trained simultaneously. It often produces superior results.

Acknowledgements

The authors would like to thank their supervisor, András Lőrincz, and also Áron Fóthi for their guidance and helpful comments. The project was supported by the European Union and co-financed by the European Social Fund (EFOP-3.6.3-16-2017-00001).

References

[1] Z. Cao, T. Simon, S-E. Wei, and Y. Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. In *The IEEE Conference on Computer Vision and Pattern Recognition*, pages 1302–1310, 2017.

[2] R. Dabral, A. Mundhada, U. Kusupati, S. Afaque, and A. Jain. Structure-aware and temporally coherent 3d human pose estimation. arXiv:1711.09250, 2017.

[3] J. F. Hu, W. S. Zheng, J. Lai, and J. Zhang. Jointly learning heterogeneous features for rgb-d activity recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(11):2186–2200, Nov 2017.

[4] C. Ionescu, D. Papava, V. Olaru, and C. Sminchisescu. Human3.6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(7):1325–1339, jul 2014.

[5] I. Laina, C. Rupprecht, V. Belagiannis, F. Tombari, and N. Navab. Deeper depth prediction with fully convolutional residual networks. In *The Fourth International Conference on 3D Vision*, pages 239–248, 2016.

[6] J. Martinez, R. Hossain, J. Romero, and J. J. Little. A simple yet effective baseline for 3d human pose estimation. In *The IEEE International Conference on Computer Vision*, pages 2659–2668, 2017.

[7] D. Mehta, H. Rhodin, D. Casas, P. Fua, O. Sotnychenko, W. Xu, and C. Theobalt. Monocular 3d human pose estimation in the wild using improved cnn supervision. In *The Fifth International Conference on 3D Vision*, 2017.

[8] D. Mehta, S. Sridhar, O. Sotnychenko, H. Rhodin, M. Shafiei, H.-P. Seidel, W. Xu, D. Casas, and C. Theobalt. Vnect: Real-time 3d human pose estimation with a single rgb camera. *ACM Transactions on Graphics*, 36(4):44, 2017.

[9] A. Newell, K. Yang, and J. Deng. Stacked hourglass networks for human pose estimation. In *European Conference on Computer Vision*, pages 483–499, 2016.

[10] A. Shahroudy, J. Liu, T.-T. Ng, and G. Wang. Ntu rgb+d: A large scale dataset for 3d human activity analysis. In *The IEEE Conference on Computer Vision and Pattern Recognition*, pages 1010–1019, 2016.

[11] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor segmentation and support inference from rgbd images. In *European Conference on Computer Vision*, pages 746–760, 2012.

[12] J. Wang, Z. Liu, Y. Wu, and J. Yuan. Learning actionlet ensemble for 3d human action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(5):914–927, May 2014.

[13] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe. Unsupervised learning of depth and ego-motion from video. In *The IEEE Conference on Computer Vision and Pattern Recognition*, pages 6612–6619, 2017.

Towards the understanding of object manipulations by means of combining common sense rules and deep networks

Máté Csákvári, András Sárkány

Abstract: Object detection on images and videos improved remarkably recently. However, state-of-the-art methods still have considerable shortcomings: they require training data for each object class, are prone to occlusions and may have high false positive or false negative rates being prohibitive in diverse applications. We study a case that a) has a limited goal and works in a narrow context, b) includes common sense rules on '*objectness*' and c) exploits state-of-the art *deep detectors* of different kinds. Our proposed method works on an image sequence from a stationary camera and detects objects that may be manipulated by actors in a scenario. The object types are not known to the system and we consider two actions: "taking an object from a table" and "putting an object onto the table". We quantitatively evaluate our method on manually annotated video segments and present precision and recall scores.

Keywords: computer vision, class-agnostic object detection, common sense, optical flow, unsupervised

Introduction

Objects appearing and disappearing is a natural part of environment dynamics. It takes place in space and time and for some tasks it is neccessary to keep track of them. For example passing of an object indirectly, when one person places and object onto a surface then another person takes it can be described as a sequence of object appearance and disappearance. The problem is that a general object definition would quickly lead to combinatorical explosion. We must find a more restrictive definition that we can handle. In this case, we consider everything that can be moved by a person with his/her hand, an object. Our base assumption is that an object does not move by itself. If there are no other motions in a scene then no object appearance or disappearance can take place. Therefore it must be a result of an actor acting upon the environment. Taking an object means, that it must be grabbed first, so there must have been some motion before and after that. Finding these points in time is our first task. We restrict ourselves to a fixed environment, namely actors placing and taking objects from a table. We propose an image difference based change detection algorithm. The simplest case is when the only difference in the images is the object. Taking the normalized image difference gives us the object that appeared or disappeared. Of course this ideal case rarely occurs. Our goal is to find images and filter them so that their difference is only the changing object.

Method

In our method's core lies the idea that, unless many interactions are happening at once, an appearing/disappearing object can be found by taking the image difference of two specific frames from a video of a stationary camera. We specify common sense assumptions and derive algorithmic components to:

1. select the two relevant frames
2. process the images to neglect irrelevant differences caused by interfering actions in the scene
3. take the image difference

Image selection

In this step we select two frames, I_{t_1} and I_{t_2}, that will be used at the image differencing step. We restrict ourselves to find objects that were moved by an actor. First, we find a point in time when the object was grabbed or put down, then search backwards and forward in time to find frames where the object is not fully occluded, to find I_{t_1} and I_{t_2}. This is done by simply checking if the bounding boxes of the hand moved significantly since the frame in which the object was grabbed.

We find timestamps of object grabbing by assuming that this action requires that the hand stops for at least an instant. Thus we look for such changes in the speed of the hand which we measure by optical flow. For optical flow estimation we use FlowNet2.0 [3]. If the magnitude of average velocity is at a

local minima, then the timestamp is selected as a candidate for change detection. The local minimas are found by using median filter on the velocity magnitude signal and then using a peak finding algorithm [2]. An example of this can be seen on Figure 1. For each of the candidate position we assume that the appearing/disappearing object is occluded by the hand in the instant of releasing/grabbing and we select a rectangular region of interest (RoI) on the image around the center of the hand. Then we search for frames backward and forward on which the hand moved outside of our selected RoI, so the object is not occluded.

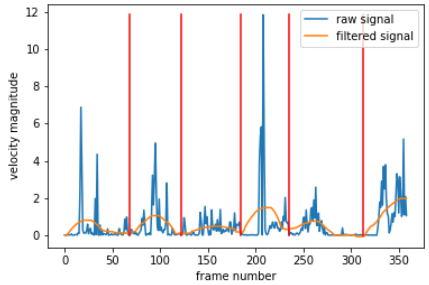


Figure 1: Hand movement segmentation. The plot shows the raw velocity magnitude filtered by median filter. The red lines are the local minimas found by the peak detector. Those timestamps will be investigated further in the next steps.

Interference removal

We found that the selected frames and RoI given by the first step contains other differences than the object we were looking for. These differences come from different sources: effect of actors interfering in the RoI (e.g. body parts, shadow, other manipulated object). To neutralize these effects we create a binary mask on the RoI that neglects pixels that belongs to these phenomena. In fact we estimate multiple binary masks with different strategies and and take the intersection of the relevant pixels found by each method. These are the following:

- Hand occlusion: We used Mask-RCNN [1] for filtering out hand-occluded pixels
- Forward and backward optical flow between I_{t_1} and I_{t_2}
- Optical flow between I_{t_1} and I_{t_1-1}, and also between I_{t_2} and I_{t_2-1}

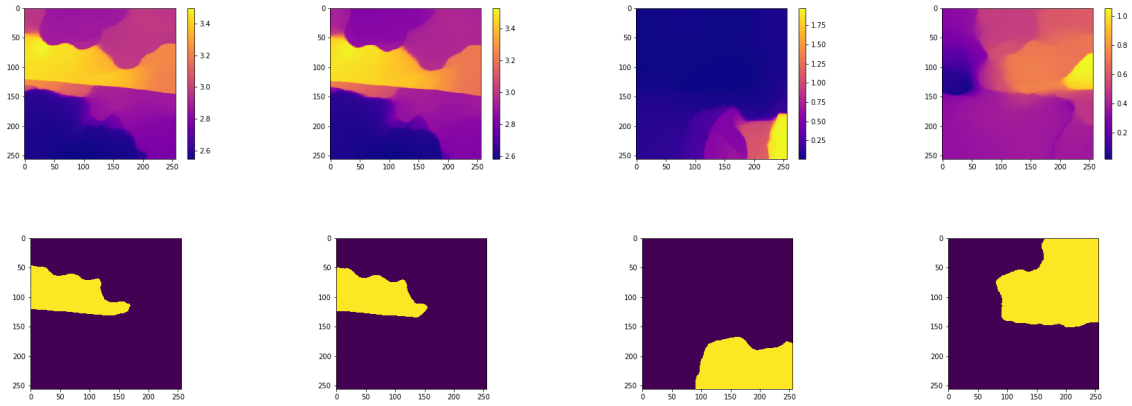
The forward and backward optical flow between I_{t_1} and I_{t_2} accounts for changes that happened over a longer period of time (e.g. edge of the paper moved on which the object was placed). The optical flow between a frame at time instant t and $t-1$ helps in removing any on-going activities in the RoI (e.g. hand is still there but moving). This could be done for any t and $t-k$ time instants, however we found $k=1$ to be sufficient. Optical flow based masks are obtained by thresholding the flow magnitude in each flow field.

Image difference

The final result is obtained by applying the binary masks on the two selected images I_{t_1} and I_{t_2} then taking their difference as follows[4]:

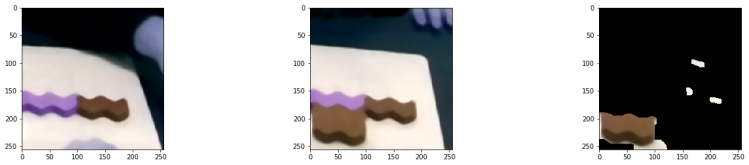
$$I_{final}(x,y) = \|I_{t_1}(x,y) - (\frac{\sigma_1}{\sigma_2}(I_{t_2}(x,y) - \mu_2) + \mu_1)\|$$

where σ_1, μ_1 and σ_2, μ_2 are the mean and standard deviation of I_{t_1} and I_{t_2} respectively. We then threshold I_{final} to detect appearance or disappearance.



(a) Optical flow between I_{t_1} and I_{t_2} (b) Optical flow between I_{t_2} and I_{t_1} (c) Optical flow between I_{t_1} and I_{t_1-1} (d) Optical flow between I_{t_2} and I_{t_2-1}

Figure 2: Optical flow magnitude (upper) and their thresholded binary masks (lower).



(a) Selected image I_{t_1} (b) Selected image I_{t_2} (c) Final mask applied to I_{t_2}

Figure 3: An example from our evaluations. The selected frames I_{t_1} and I_{t_2} during image selection and the objects that appeared between the two time instants as found by our method.

Detector left out	**Precision**	**Recall**
Optical flow between I_{t_1}, I_{t_2}	0.76	0.47
Optical flow between I_{t_1}, I_{t_1-1} and I_{t_2}, I_{t_2-1}	0.71	0.42
Mask-RCNN	0.72	0.52
All detectors active	**0.82**	**0.55**

Table 1: Results of leave-one-out experiments for the different detectors.

Results

We evaluated our method on a dataset provided by Argus Cognitive Inc. It containts sessions where two people are interacting over a table with a static camera above them. We labeled time intervals when the object was grabbed but did not start to move. Finding one point in this interval counts as a true positive detection. We suppressed multiple detections by filtering out points that are closer than the length of the shortest interval in our dataset. The algorithm achieved 82% precision with 55% recall. We did leave-one-out experiments to determine the relevance of each detector based binary mask. Table 1 shows the result of this experiment.

Conclusion

We proposed a method for detecting appearing and disappearing objects without the use of training samples. First by describing the general driving principles of the process, then transforming them into concrete rules. We used high accuracy detectors for each rule which resulted in acceptable overall perfomance.

While this method has acceptable performance, many improvements can be made. The motion segmentation step greatly affects the performance, since it proposes the candidates for detection. Improving this step would be highly beneficial. Another area to investigate further is the treatment of hyperparameters. At almost every step of this method there are thresholding parameters which require careful tuning. These include the bounding box difference threshold in the forward-backward image search for candidate images, the optical flow thresholding parameters and the bounding box size for hand motion. While we can find generally good values for these, treating them in a probabilistic fashion would be of great importance. We could then measure the uncertainty in both the parameters and the detection. This latter topic will be explored in the future.

Acknowledgements

We couldn't have done this work without the great advices from our supervisor Dr. habil András Lőrinc and also from Zoltán Tősér. We would also like to thank Dr. Erzsébet Csuhaj Varjú as professional leader. The project has been supported by the European Union, co-financed by the European Social Fund EFOP-3.6.3-16-2017-00002.

References

[1] He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017, October). Mask r-cnn. In Computer Vision (ICCV), 2017 IEEE International Conference on (pp. 2980-2988). IEEE.

[2] Du, P., Kibbe, W. A., & Lin, S. M. (2006). Improved peak detection in mass spectrum by incorporating continuous wavelet transform-based pattern matching. Bioinformatics, 22(17), 2059-2065.

[3] Ilg, E., Mayer, N., Saikia, T., Keuper, M., Dosovitskiy, A., & Brox, T. (2017, July). Flownet 2.0: Evolution of optical flow estimation with deep networks. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (Vol. 2).

[4] İlsever, M., & Ünsalan, C. (2012). Pixel-based change detection methods. In Two-Dimensional Change Detection Methods (pp. 7-21). Springer, London.

Visualization of test-to-code relations to detect problems of unit tests

Nadera Aljawabrah, Tamás Gergely

Abstract: Visualization of test information can be a means of providing an understanding of test-code relations which, in turn, is essential for other activities in SDLC such as: maintenance, software evolution, refactoring, etc. To our best knowledge, less research have specifically addressed the visualization of test-code relation and its importance in many tasks during software development. This paper is a modest contribution to draw attention to visualization of test-code relations and in particular visualizing of test-to-code traceability links. From the outcome of our investigation it has been demonstrated that visualizing of test-related metrics has been received rather interest of researchers; while in turn, little attention has been paid to visualizing of test to code traceability links. However, several practical questions emerged which can be investigated as the next step of our research , thus further research of visualization of test-to-code traceability links area is highly needed to address these questions.

Keywords: Software testing; traceability recovery; test-to-code traceability; visualization

Introduction

Testing is considered an important phase in software development life cycle (SDLC) in which tests play a significant role in software evolution and maintenance, and helps building a better quality product. To achieve this, it is important to understand the tests and how they are related to the production code.This knowledge is essential for several activities such as: change impact analysis, refactoring, re-engineering, etc.

Visualization in general is widely and effectively used in software development for different purposes as it helps understanding. It can also be a very effective method to support testers to quickly understand the goal and properties of each test, to overview the structure of the test suite and to recognize the relation between test and code items. Many publications have been investigated in the domain of software visualization, source-code related visualization and test related visualization. This work considers visualization of relations between test and code from two aspects: visualization of test information and visualization of test-to-code traceability links.

Visualization of test information can be a means of providing a valuable information about: the adequacy of code testing [11], visualizing software faults [4], and code coverage of test suites to evaluate their quality [1]. The area of test information visualization has been widely targeted. For example, Cornelissen et al. [3] presented a visualization approach based on UML sequence diagram to visualize test information. While Jones et al. [6] proposed a technique that visualizes all statements that are executed by test suites and facilitates the location of faults in these statements. Quite recently, there has been a growing interest in visualizing test-related metrics which could provide information about the quality and process of the tests and the testing effort. However, to the author's best knowledge, very few publications are available in the kinds of literature that address the issue of visualizing the relationship between tests and code under test in term of test-to-code traceability links.

Our research takes into consideration two test-code relation areas: Test-related metrics and test-to-code traceability links. In this work, we define open questions to be answered in future research. The questions address the visualization of test-to-code traceability links from different aspects, for instance, why to visualize test-to-code links, how we can visualize links, what else could be visualized, and so on. By using visualization we could check, for example, the degree of consistency among traceability links, thus it becomes easier to understand how the test and code are really related and to determine if something is going wrong in this relation.

The remainder of this paper is organized in different sections as follow. Test related visualizations are detailed in Section , followed by open research area in Section , we conclude in Section .

Test-code relations visualization

Test-related metrics has been gaining importance in visualization's realm in recent years. Code coverage metrics is one of the most popular test metrics and is used to measure the percentage of code that

is executed by the test cases. There are several tools have been developed to visualize code coverage. Balogh et al. extended CodeMetropolis to include visualization of test related metrics in the Minecraft world to support developers to better understand the test suites quality and its relation to the production code [2]. Different types of test metrics are determined to show the behavior of test on the code of different units; e.g. code coverage metrics, partition metrics and other specific metrics. In the visualization space, code units are represented as buildings protected by outposts which, in turn, represent test suites. The attributes of the outposts reflect the quality attributes of the tests.

Visualization of test suites is useful to give any reader an obvious view of the testing results as well as it can have a significant effect on the quality of the software by reducing the cost and effort required to locate faults in the source code. On the other hand, establishing links between units under test and its related test suites helps in reducing regression tests and in turn saving much time during software development and maintenance [10]. Traceability links could be identified using: manual, automatic, semi-automatic or explicit methods [7]. Various approaches have been proposed to retrieve the traceability links between the tested units and units under test. Rompaey and Demeyer [12] compared six traceability strategies in terms of the applicability and the accuracy of each approach. The comparison covered only those approaches relating to requirement traceability and test-to-code traceability: Naming convention (NC), Fixture Element Types (FET), Last call before assert (LCBA), Lexical Analysis (LA), Static Call Graph (SCG) and Co-evolution (Co-Ev). According to these approaches, units under test are identified by matching test cases and production code's names and vocabulary (e.g. identifiers, comments), examining method invocations in test cases, looking at the last calls right before assert statement, and capturing changes on test cases and production code in the version control change log [12]. These approaches have some limitations particularly in industrial projects with practical usability or support. For instance, there are no industrial tools available to support automatic test-to-code traceability. In addition, visualization test-code links are not supported by any of above approaches [8] despite of the importance of visualization in the areas of software maintenance, evolution and refactoring.

Most of test-to-code traceability recovery approaches retrieve high-level links, i.e. links between unit tests and classes under test [9]. An automated test-to-code traceability approach has been proposed to recover links between source code and test cases on the method level by identifying Focal method under test [5].

Most of the visualization techniques provide visualization of relationships/links between requirements and other software artifacts (source code, design, test cases), while none of these techniques aim visualization of test-to-code traceability links in specific.

Open Research area

Depending on our investigations it is clear that further research in the area of visualization of test-to-code traceability links is necessary. We provide some open questions that can help to reveal specific topics in this area for further research and try to give some example answers. For ourselves, we feel these questions and directions the most interesting ones.

- *First, what the purpose of visualization can be ?* Visualization must have a purpose. Defining our goal can help in finding proper visualization techniques to be used and appropriate elements to be presented in it. Purpose can be: understand relations, impact analysis, find problems ("bad smells").

- *What is the suitable visualization technique that can be used to display test-to-code traceability relations and their attributes?* There are several possible ways to visualize relations including graphs, matrices, hyperlinks, lists, tree maps, 3D space. Based on the existing visualization techniques addressing traceability links between various software artifacts, graph-based visualization and traceability matrices seem to be the most suitable methods to represent links between code and tests. However, this may depend on the purpose. For example, when one tries to check the relations of an item for impact analysis, graph representation,as well as hyperlinks, seem to be appropriate. On the other hand, if someone needs a broader view to check inconsistencies among the relations, graph representation showing the traceability links inferred using different link-detection techniques in different colors might be a better choice. But a 3D visualization also seems to be appropriate to show attributes of different items and relations.

- *What test and code items, and properties should be objects and attributes in the visualization?* Relations can be visualized directly as objects in the visualization space (e.g. as lines between objects), or we can only map their properties to the attributes of the connected objects.
- *What the criteria should be taken into account to choose visualization technique?* For instance, the size of the program can be taken into account as several visualization methods can often become large and thus hard to read and understand for big projects.
- *What is the best recovery approach usable to retrieve the links between test and code?* There are several techniques can be used to infer these traceability relations, and each technique retrieves a slightly different set. Depending on the purpose of the visualization and the technique we use, either all can be visualized or we should choose and build on one of them. But which one? This is another open question that can be investigated.
- *At what detail level information should be visualized?* In a real system, there are thousands of tests and code items exist. Although it is not impossible to visualize all these at once, it is probably not the best way (although not necessarily impossible). Instead, a selective or hierarchical visualization seems to be a better choice. For example, instead of method level, one can show (test and production) classes, or group items based on their relations or some other purposes and visualize the groups only.

Conclusions

From the research that has been carried out, it is possible to conclude that more research in the visualization of test-to-code traceability is quite necessary. As there are many sources from where the traceability relations can be inferred, one of the most important questions is to decide which source or combination of sources are the best to determine the test-to-code links. It is obvious, that if these sources disagree, this will make it harder to understand what is going on, what was the goal of the developer, how the components are really related, change impact analysis can yield in false results, etc. Fortunately, visualization can aid this task. The goal of using visualization in this case is to "see" the disagreement between traceability links inferred from different sources. This might point out places where something is wrong with the tests and/or the code (at least their relationship) in a specific system. Further research then can aim the statistical comparison of these inference methods. Working on addressing these questions is continuing and will be presented in future papers.

References

[1] Vanessa Peña Araya. Test blueprint: an effective visual support for test coverage. In *Proceedings of the 33rd International Conference on Software Engineering*, pages 1140–1142. ACM, 2011.

[2] Gergo Balogh, Tamás Gergely, Arpád Beszédes, and Tibor Gyimóthy. Using the city metaphor for visualizing test-related metrics. In *Software Analysis, Evolution, and Reengineering (SANER), 2016 IEEE 23rd International Conference on*, volume 2, pages 17–20. IEEE, 2016.

[3] Bas Cornelissen, Arie Van Deursen, Leon Moonen, and Andy Zaidman. Visualizing testsuites to aid in software understanding. In *Software Maintenance and Reengineering, 2007. CSMR'07. 11th European Conference on*, pages 213–222. IEEE, 2007.

[4] Marco D'Ambros, Michele Lanza, and Martin Pinzger. "a bug's life" visualizing a bug database. In *Visualizing Software for Understanding and Analysis, 2007. VISSOFT 2007. 4th IEEE International Workshop on*, pages 113–120. IEEE, 2007.

[5] Mohammad Ghafari, Carlo Ghezzi, and Konstantin Rubinov. Automatically identifying focal methods under test in unit test cases. In *Source Code Analysis and Manipulation (SCAM), 2015 IEEE 15th International Working Conference on*, pages 61–70. IEEE, 2015.

[6] James A Jones, Mary Jean Harrold, and John Stasko. Visualization of test information to assist fault localization. In *Proceedings of the 24th International Conference on Software Engineering*, pages 467–477. ACM, 2002.

[7] Andrian Marcus, Xinrong Xie, and Denys Poshyvanyk. When and how to visualize traceability links? In *Proceedings of the 3rd international workshop on Traceability in emerging forms of software engineering*, pages 56–61. ACM, 2005.

[8] Reza Meimandi Parizi, Asem Kasem, and Azween Abdullah. Towards gamification in software traceability: Between test and code artifacts. In *Software Technologies (ICSOFT), 2015 10th International Joint Conference on*, volume 1, pages 1–8. IEEE, 2015.

[9] Reza Meimandi Parizi, Sai Peck Lee, and Mohammad Dabbagh. Achievements and challenges in state-of-the-art software traceability between test and code artifacts. *IEEE Transactions on Reliability*, 63(4):913–926, 2014.

[10] Abdallah Qusef. Test-to-code traceability: Why and how? In *Applied Electrical Engineering and Computing Technologies (AEECT), 2013 IEEE Jordan Conference on*, pages 1–8. IEEE, 2013.

[11] Thomas Tamisier, Peter Karski, and Fernand Feltz. Visualization of unit and selective regression software tests. In *International Conference on Cooperative Design, Visualization and Engineering*, pages 227–230. Springer, 2013.

[12] Bart Van Rompaey and Serge Demeyer. Establishing traceability links between unit test cases and units under test. In *Software Maintenance and Reengineering, 2009. CSMR'09. 13th European Conference on*, pages 209–218. IEEE, 2009.

Parallelisation of Haskell Programs by Refactoring

Norbert Luksa, Tamás Kozsik

Abstract: We propose a refactoring tool for the Haskell programming language, capable of introducing parallelism to the code with reduced effort from the programmer. Haskell has many ways to express concurrency and parallelism. Moreover, Eden, a dialect of Haskell supports a wide range of features for parallel and distributed computations. After comparing a number of possibilities we have found that the Eval Monad, the Par Monad and the Eden language provide similar parallel performance. Our tool is able to introduce parallelism by turning certain syntactic forms into the application of algorithmic skeletons, which are implemented with the Eval Monad, the Par Monad and the Eden language.

Keywords: Parallel Programming, Haskell, Eden, Par Monad, Eval Monad, Refactoring, Haskell-tools

Introduction

When it comes to programming, there are various factors to keep in mind. First of all, the program should accomplish the given task, but in the meantime it should be also efficient, furthermore the code should be clean and readable. Focusing on the speed, we have several possibilities for improvement. We often try to write efficient programs, which the compiler can further enhance with optimizations, but after a point there is no way to speed up a sequential program: this is when parallelism comes to mind.

The *Haskell* programming language has a wide range of possibilities to express parallelism. We have analyzed some of these methods, such as the *Eval Monad* with its parallel evaluation strategies, the *Par Monad*, and *Eden*, an extension of *Haskell*, a language with constructs for the definition and instantiation of parallel processes. Our goal was to find the best among these not only in terms of speed up, but also while looking at the ways these methods support parallelism – hence our final goal is to turn a sequential program to parallel with a refactoring tool, with as little effort from the programmers' side as possible.

Options for Parallelism

As part of our research we have examined various methods to describe parallelism. From these, we have compared three approaches in more detail through case studies of different complexity.

Evaluations Strategies

Pure parallelism in *Haskell* is achieved using only two primitives:

```
1 par  :: a -> b -> b
2 pseq :: a -> b -> b
```

While `par` indicates that it may be beneficial to lazily evaluate the first argument in parallel with the second, `pseq` forces this evaluation to actually take place. Although they are not too convenient to use, with an extra layer of abstraction they suddenly become a great and simple tool to express parallelism. This is achieved by introducing the simple idea of *Evaluation Strategies*, or *Strategies* for short [3]. A Strategy is a function with the following simplified type:

```
1 type Strategy a = a -> Eval a
2 data Eval a = Done a
3 runEval :: Eval a -> a
```

Then we can define some basic strategies, like the monadic counterparts of the functions above, and some more complex ones, such as `parList` – which evaluates the elements of a list in parallel –, and many others. There is a great thing with these Strategies when we think about refactoring: a simple construct lets us turn a sequential program into parallel without changing anything else in the code. The example below shows how to create a function, which would map a function over a list in parallel:

```
1 using :: a -> Strategy a -> a
2 x `using` strat = runEval (strat x)
3 parMap f xs = map f xs `using` parList rseq
```

The Par Monad

Another useful tool is the Par Monad [4], which is built around the concept of *forking*: the computation passed as the argument to function `fork` is computed in parallel. We can see this as a *parent-child* evaluation, where the passed argument is computed in the child process, while the current computation is executed in the parent process.

```
1 fork :: Par () -> Par ()
```

Adding once again an extra layer over this simple basic construct, we can have another great tool. The communication among the processes is accomplished through `IVar`s (I-structures), which are write-once mutable reference cells with the operations `put` and `get`. Function `put` assigns a value to an `IVar`, while `get` returns the value after the assignment takes place (and blocks until this happens).

```
1 new :: Par (IVar a)
2 get :: IVar a -> Par a
3 put :: NFData a => IVar a -> a -> Par ()
```

By defining a common pattern that creates a child and then collects the result, `spawn`, we can easily give the definition of `parMap` again, now using the Par Monad.

Eden

Eden is a parallel functional language, extending Haskell with constructs for creating and computing processes in parallel [1].

```
1 process :: (Trans a, Trans b) => (a -> b) -> Process a b
2 (#)     :: (Trans a, Trans b) => Process a b -> a -> b
```

Here `process` creates the *process abstraction*, while `(#)` instantiates it. The evaluation of an expression like `(process func) # arg` leads to the evaluation of the argument `arg` on a new thread in the parent process and will be sent to the child process which returns the result of `func arg` in a demand driven way. The type context `Trans a` means that `a` has to be *transmissible*. The two constructs described above are usually used in a combination resulting in parallel function application:

```
($#) :: (Trans a, Trans b) => (a -> b) -> a -> b
f $# x = process f # x
```

So once again we have the basic construct – but we can add an extra layer of abstraction to have a convenient way of writing a parallel program.

Comparison

We have tested the performance of the above mentioned approaches on simple and more complex problems [2, 5]. The achieved speed-up lived up to our expectations with all three approaches showing good results. We also had to consider other factors as well, most importantly how complex it is with the three approaches to introduce parallelism as code transformations to a sequential program.

After several case studies, we have found the three different methods equally convenient, so we decided to support all the three. However, since *Eden* and its predefined algorithmic skeletons provided a very nice abstraction layer, we have implemented these skeletons using the other two approaches in order to provide a common ground for the parallelisation code transformations.

General skeletons

To generalize over the three considered parallelization methods, we have introduced general algorithmic skeletons. We have decided to define these skeletons in a language-independent manner. Our restriction only lies in the necessary methods, such as *map* or *parMap*. Given these functions, we can specify different algorithms in a general way on any traversable structures.

Function *parMap* can be considered the simplest skeleton, which takes a list and creates a process executed in parallel for each of its elements. Of course, this naive version cannot be used in a general problem where the length of a list is probably larger, large enough that the communication among the processes would result in severe overhead. An improvement in our case is the *farm* skeleton which distributes its input to a number of worker processes, just like in the *Eden* version. For this, we need two new functions: one which distributes the elements, and another one which combines the results. Traversable data structures will be denoted with curly braces, such as `{Int}`.

$$\begin{aligned} Distribute &\triangleright \{a\} \to \{\{a\}\} \\ Combine &\triangleright \{\{a\}\} \to \{a\} \end{aligned}$$

$$\frac{Combine \circ Distribute \equiv Id, \quad f \triangleright a \to b}{Farm(f) \equiv Combine \circ ParMap(Map(f)) \circ Distribute}$$

Observe that we can allow the inside and outside structures in the domain of *Combine* and in the co-domain of *Distribute* to be different: we can split a tree into a list of trees, for instance.

There are quite a few common parallel patterns which we can express with the general skeletons. We have given definitions for reduction, for map-reduce, and for divide-and-conquer schemes.

Transformation to expressions

For each of our general skeletons we have defined their transformations to abstract expressions. At this stage we have still stayed with the general-purpose solution, but we have also kept in mind that we want to introduce code transformations in the *Haskell Tools* refactoring environment [6]. Therefore we use expressions as represented in this tool. In order to have a better understanding of these expressions, consider the following example.

$$\begin{aligned} 3+2 &\triangleright Expr \\ 3+2 &\triangleright Expr \to Expr \to Expr \to Expr \end{aligned}$$

Here $3 + 2$ can be considered a simple expression, or we can look at it as a combination of three expressions creating a new one. However, if we introduce the pattern of *Operator* on the $(+)$ sign, we can see the type signature of an infix application. If we introduce its pattern, we can give the type of the addition as follows.

$$\begin{aligned} InfixApp &\equiv Expr \to Operator \to Expr \to Expr \\ 3+2 &\triangleright InfixApp \end{aligned}$$

We have defined the syntactic occurrence of algorithmic skeletons using the Haskell Tools' representation of Abstract Syntax Trees.

Refactoring

In order to provide a convenient way to refactor sequential Haskell programs into parallel, we have extended *Haskell Tools*, a refactoring environment for *Haskell* programs [6] with support for parallelization transformations.

Transforming a sequential code to parallel is just the first step. Ensuring that the parallel code runs faster than the sequential one is also difficult. The costs of process creation and communication may cancel out the performance gains of parallel evaluation. A tool should help the developer make decisions on which parallel skeleton, if any, to use in a given situation to achieve significant efficiency improvement. Cost-directed refactoring [7, 8, 9] uses cost models to facilitate such decisions – our future work will target this approach.

Acknowledgement

Norbet Luksa was supported by ÚNKP-17-1 New National Excellence Program of The Ministry of Human Capacities (Hungary). Tamás Kozsik was supported by the European Union, co-financed by the European Social Fund (EFOP-3.6.2-16-2017-00013)

References

[1] Loogen, R. (2011). Eden – Parallel Functional Programming with Haskell. In: *Proc. 4th Central European Functional Programming School*, pp. 142–206. Springer

[2] Marlow, S. (2011). Parallel and Concurrent Programming in Haskell. In: *Proc. 4th Central European Functional Programming School*, pp. 339–401. Springer

[3] Marlow, S., Maier, P., Loidl, H., Aswad, M., Trinder, P. (2010). Seq no more: better strategies for parallel Haskell. In: *Proc. 3rd ACM Symposium on Haskell*, pp. 91–102. ACM

[4] Marlow, S., Newton, R., Jones, S.P. (2011). A monad for deterministic parallelism. *SIGPLAN Not.* **46**(12):71–82.

[5] Loidl, H., Totoo, P. (2014). Parallel Haskell implementations of the N-body problem. *Concurrency and Computation: Practice and Experience* **26**:987–1019.

[6] Németh, B. *Haskell-tools*. Retrieved from https://github.com/haskell-tools/haskell-tools

[7] Brown, C., Danelutto, M., Elliott, A., Hammond, K., Kilpatrick, P. (2013). Cost-Directed Refactoring for Parallel Erlang Programs. *International Journal of Parallel Programming* **42**:564–582.

[8] Berthold, J., Hammond, K., Loogen, R. (2003). Automatic Skeletons in Template Haskell. *Parallel Processing Letters* **13**:413–424.

[9] Loogen, R., Ortega, Y., Peña, R., Priebe, S., Rubio, F. (2003). Parallelism abstractions in Eden. In: *Patterns and Skeletons for Parallel and Distributed Computing*, pp. 95–128. Springer

JavaScript-only Parallel Programming of Embedded Systems

Péter Gál

Abstract: The number of Internet-connected small embedded devices are increasing rapidly. Traditionally, such devices are programmed in some kind of compiled language, e.g., in C or C++. However, the number of embedded developers is small in contrast to JavaScript developers. Fortunately, there are multiple projects, which are providing JavaScript engines for such small devices and allow JavaScript developers to also program embedded devices. Nowadays, the embedded devices are not necessarily single-core systems anymore but can have multiple cores. For multi-core CPUs, the way to execute tasks or processes in parallel is a must. However, the JavaScript language itself does not have any thread, task, or parallelism concepts.

In this work, a way to run JavaScript scripts in parallel on embedded systems is presented. The proposed implementation adopts an already existing W3C standard as starting point to provide a familiar API for Web developers, and it is extended with properties required for embedded devices. By using the Worker concept, the embedded JavaScript programmer can leverage multi-core systems and also execute various tasks in separate contexts which do not block the main JavaScript execution.

Keywords: Embedded systems, JavaScript, Parallel programming, Workers

Introduction

The popularity of JavaScript is ever growing since it is was first used on the Web. Stack Overflow reports that at least 60% percent of developers are using JavaScript [7]. The most popular language on GitHub is also JavaScript [6]. In contrast, there are far less embedded developers reported by these surveys. However, the increasing number of Internet-connected devices in the Internet of Things movement results in a requirement for more embedded programmers. By enabling the use of JavaScript in embedded software or software components, JavaScript developers can contribute to the development of IoT devices.

Another interesting point in embedded systems is that there are more and more multi-core CPUs present in the embedded hardware segment. Using various parallel programming techniques, the embedded applications could benefit from the multi-core CPUs. Usually the concept of tasks or threads is available in embedded operating systems. FreeRTOS uses the task concept where a method can be started as a task and the OS schedules the execution of all tasks [2]. The ARM mbed OS uses the threading concept to enable parallel execution [3]. Currently, in the JavaScript language specification, there is no task or thread concept which would allow parallel execution.

In this paper, the current state of JavaScript parallel execution is investigated and a solution is presented for embedded devices.

Background

Running JavaScript on embedded devices is not an impossible task. There are already multiple engines designed for small devices, e.g., Espruino [1], Duktape [8], and JerryScript [5]. Each of these engines are created to have small memory footprint (both ROM and RAM) and to be as ECMAScript 5.1 compatible as possible.

The ECMAScript 2015 Language Specification introduces the Promise API for JavaScript [4]. The Promise objects represent a series of asynchronously invoked JavaScript functions, each of which is enqueued as job which needs to be executed by the JavaScript engine's event loop. As these functions are executed in a serial fashion by the event loop, two Promise functions cannot actually run in parallel. This means that this approach can not achieve true parallelism. The Promise object is available in modern browsers and also in the Node.js framework.

In Node.js, there is a possibility to execute JavaScript code in a child process. This is achieved by exposing the exec/frok mechanism of the OS to the JavaScript runtime. Child processes can communicate with their parent process via IPC channels via message passing. Using the child processes on non-embedded computers, parallelism can be achieved as each process runs separately. However, on

embedded devices it is highly possible that the OS does not support the creation of child processes. FreeRTOS and Mbed OS are such operating systems.

Fortunately the World Wide Web Consortium (W3C) have created a standard to allow executing JavaScript code on a separate thread. This is the Web Workers specification [9], which defines an API that allows web application developers to spawn background workers running JavaScript in parallel to their main page. Workers have no direct access to the main page contents but have means of communication via message passing. Values and objects sent in the messages are copied and not shared, therefore there are no synchronization issues while accessing data from multiple threads.

This Web Worker approach is a good base concept for the parallel execution of JavaScript code on embedded devices as it can be mapped to the task or thread concept of embedded OSs. For example, each new Worker can be expressed as the creation of a task and running the required code in that new context. Moreover, the Web Worker API is a technique well-known to JavaScript developers familiar with web-related programming, thus it enables the transfer of their knowledge to the embedded domain. This solution is presented in more detail in the following section.

Workers for Embedded

The Web Worker standard describes the `Worker` JavaScript objects which can be used to initialize and execute additional JavaScript contexts. To instantiate a Web Worker, only the script source is required as its first argument. At the time of the creation of the new context the Worker will automatically start. This can be observed in the line 1 of Listing 1. To send data to a newly created Web Worker, the worker object's `postMessage` method can be used, as depicted on line 5. The method serializes its parameter data and sends it to the worker's context. As the data is serialized, each JavaScript context will have a copy of the value. Only the `postMessage` method can be used to send data to the Web Workers. Additionally, the `postMessage` method in the Web Worker's context can send data to the parent context. Messages sent via the `postMessage` can be handled by implementing the `onmessage` method, both on the main script's worker object and in the new worker's context. The message handler method receives an `event` argument which describes message received. In this `event` object, the `data` property stores the content sent by the `postMessage` method. The data received this way is deserialized and can be treated as a normal JavaScript object or value. An example on how a message can be received from a worker is presented on lines 2 and 3 in Listing 1. It is important to note that the concrete serialization and deserialization mechanism is not mentioned in the standard but is left as an implementation-defined process instead. A Web Worker will continue to wait for messages till the parent context invokes the `terminate` method on the worker object. This call is omitted from the examples.

These Web Worker mechanisms can be mapped to traditional embedded programming techniques. The creation of worker is the task creation. Communication via the `postMessage` and `onmessage` methods is similar to message queue operations in embedded devices. This is the main reason why the Web Worker concept is ideal for the embedded JavaScript world. Additionally it is important to mention that the Worker context is a separate one which does not block the main context.

```
1 var work = new Worker('script.js');
2 work.onmessage = function(event) {
3  // event.data
4 }
5 work.postMessage(10);
```

Listing 1: Example Web Worker initialization

Proof-of-Concept Implementation

A proof-of-concept implementation was created using the JerryScript [5] JavaScript engine and the FreeRTOS system [2]. For each new Worker created in JavaScript, an underlying FreeRTOS task is created. At the construction state, the script which should be executed will be associated with the worker object. As embedded systems do not necessarily have access to file systems, the filename of the given JavaScript filename will be resolved internally using a filename-source code mapping generated during the compilation of the embedded software.

After the new Worker object is created in the current JavaScript context, it is ready to be started. In the implementation, the Worker can be started with the `start` method. Upon starting the Worker the underlying task will first create a new JavaScript context to separate all methods and variables from the main JavaScript context. After this, the Worker's JavaScript code will be executed and the task enters an event loop state waiting for messages from the main JS context and from the worker context.

The underlying communication between the tasks in FreeRTOS is created via Queues. If the parent JavaScript context sends a message using the `postMessage` method to the child worker, underlying implementation sends a message to the task's own Queue. On any new message to the Queue, the worker's event loop wakes up and processes the entry on the Queue by sending the message to the `onmessage` event handler method. After handling the message, the event loop enters into a sleeping state waiting for furher messages. The message from the child worker to the parent one is done in the similar fashion.

An example main JavaScript and a worker script is shown in Listings 2 and 3. In the example, the `job.js` file is responsible for calculating the sum of the first `N` integer numbers. The `N` is provided for the script from the parent JavaScript context as an event message, as the line 2 illustrates. An input value, that is the `N` number, for the calculation can be observed on line 7 in Listing 2. After the sum is computed, the result is sent back from the worker on line 7 of Listing 3.

```
1 var work = new Worker('job.js', 120);
2 work.onmessage = function(event)
3 {
4  // event.data
5 }
6 work.start();
7 work.postMessage(10);
8 work.postMessage(30);
```

Listing 2: Main JavaScript

```
1 onmessage = function(event) {
2  var sum = 0, N = event.data;
3  for (var i = 0; i < N; i++)
4  {
5   sum += i;
6  }
7  postMessage(sum);
8 }
```

Listing 3: An example job.js JavaScript

Embedded Extensions for Workers

The API of the Web Workers provide minimalistic fine tuning options for a developer. However, more fine tuning is present for task executions in embedded systems. One of the more important things is the specification of the task stack size. In the embedded Worker approach, this information can be added via an additional parameter to the constructor call. This can be observed on line 1 in Listing 2 where the `Worker` function is called with the second parameter representing the allowed stack size for the new Worker task.

Another extension added to the original API is the introduction of the `start` method for worker objects. Using this method the start of a created Worker can be explicitly controlled. The start-up can be delayed if needed. This can be useful if the developer creates all required Worker objects first, then connects the message handlers, and finally starts up the required tasks.

Summary and Future Work

This paper presents a way to execute multiple JavaScripts in parallel on small embedded devices by reusing and extending an API which is already familiar to JavaScript developers: (Web) Workers. The Worker-based solution allows communication via message passing between runtime contexts. Using this approach, JavaScript developers can exploit the benefits of multi-core embedded systems. By running JavaScript on multi-core embedded system rapid prototyping is also encouraged.

There are still other Web Worker-based aspects to explore. For example, the Web Worker W3C standard describes Shared Workers which enables multiple connections between workers. Also, as the communication channels are always copying data between the contexts, the memory usage can increase rapidly, which is usually something to be avoided on embedded systems that typically have very limited memory on-board. By improving the way how messages are passed, e.g., by using shared memory, the implementation could reduce the memory impact and even potentially provide application speed-up.

Acknowledgment

This research was supported by the EU-funded Hungarian national grant GINOP-2.3.2-15-2016-00037 titled “Internet of Living Things”.

References

[1] Espruino. https://www.espruino.com/.

[2] Freertos. http://www.freertos.org.

[3] ARM. Mbed os. https://www.mbed.com/en/platform/mbed-os/.

[4] Ecma International. *ECMAScript 2015 Language Specification*. Geneva, 6th edition, June 2015.

[5] JS Foundation. Jerryscript. http://jerryscript.net/.

[6] GitHub. The state of the octoverse 2017. https://octoverse.github.com/, 2017.

[7] Stack Overflow. Developer survey results. https://insights.stackoverflow.com/survey/2017, 2017.

[8] Sami Vaarala. Duktape. http://duktape.org/.

[9] W3C. Web workers. W3c candidate recommendation, W3C, May 2012.

Primitive Enthusiasm: A Road to Primitive Obsession

Péter Gál, Edit Pengő

Abstract: Code bad smells usually indicate that there are low quality, hardly readable and maintainable parts in the source code. Static analysis tools can help programmers to identify bad smells and guide the refactoring process. Bad smell detection techniques have a considerable amount of literature although there are still a few types that need further study by the research community. The Primitive Obsession smell is one of them.

In this paper, we studied the definition of Primitive Obsession and based on that we introduced a method level metric, Primitive Enthusiasm, that can be used as an indicator for the smell. We implemented an algorithm for Primitive Enthusiasm calculation on Java code and analysed two real life Java systems along with a sample project. As no benchmark was available, we performed a manual validation of the results.

Keywords: Code smells, Primitive Obsession, Static analysis, Refactoring

Introduction

Refactoring means improving different attributes of the program without changing its external functionality [2]. Through code changes the original structure of the code will decay, making it harder to identify design elements, understand its behaviour, and to find bugs. The main goal of refactoring is usually to reduce the complexity of the code, to improve its readability, and therefore to make it more maintainable. There are several indicators that suggest the need of refactoring in the code. The code smells identified by Fowler can be such indicators [6].

Primitive obsession, that can be vaguely interpreted as the overuse of primitive data types, has lacked the attention of the research community. A recent literature review by Gupta et al. [3] concluded that currently existing smell detection tools and techniques do not provide support for this type of smell. They collected 60 research papers form 1999 to 2016 and found that four of Fowler's bad smells – including Primitive Obsession – were not detected in any of the papers.

Therefore, we decided to study Primitive Obsession and defined an indicator which can highlight the potential places where this bad smell could occur. We call this indicator Primitive Enthusiasm and it focuses on one major aspect of the smell. We implemented the indicator algorithm for Java source code, however, the method can be generalized to other languages as well. The algorithm was tested on two real life Java projects and on a sample program seeded with Primitive Obsession. A manual validation of the results was performed.

Defining Primitive Obsession

In most programming languanges, data types can either be categorised as primitive or complex types. The building blocks of complex types are primitive types, e.g., integers or booleans. The benefits of complex types is that they usually provide semantic information about the variables. For example, from three integer values we can form a meaningful structure of a date record. It is much easier to grasp the essence of a code fragment if the operations are carried out on small, straightforwardly named objects rather than using dozens of distinct primitive types.

Primitive Obsession smell means that the programmer does not create small objects for small tasks, instead s/he is obsessed with the use of primitive data types. However, this definition is not really exact and can be interpreted broadly. Chapter 3 of the Fowler book [2] makes the previous guideline clearer by providing a list of possible refactorings for Primitive Obsession. A small GitHub project [5] aids our understanding by showing these refactorings step-by-step in practice.

The code fragment in Figure 1 is packed with suspicious lines. Using type code like the constant class variables `ENGINEER` and `SALESMAN` instead of a State or Strategy class can be considered as Primitve Obsession. From the first three parameters of the `work` method, the developer could create a Shift class. A method having too many primitive data types in its parameter list can be a sign of Primitive Obsession. The extraction of class members to primitive local variables, like on lines 5 and 6, can also be recognised as Primitive Obsession.

```
1  class Employee{
2  static const int ENGINEER = 1;
3  static const int SALESMAN = 2;
4  public void work(int from, int to, int numberOfBreaks, Task task){
5  String taskName = task.getName();
6  int hardness = task.getHardness();
7  /* ... */
8  }
9  }
10
```

Figure 1: Sample code contaning three Primitive Obsessions

The sample code in Figure 1 shows that identifying Primitive Obsession usually needs semantic knowledge about the code. Moreover, the number of primitive types used in a software is application, problem, and coding guideline dependent. Thus, it is not possible to formulate an exact rule on how much the primitive types can be used before we report Primitive Obsession.

Indicator for Possible Primitive Obsession

Considering the challenges described in Section , we decided to restrict our definition of Primitive Obsession. In this section we present Primitive Enthusiasm, a metric based on this more exact definition. Primitive Enthusiasm can be used as an indicator to highlight potential candidates for Primitive Obsession.

Implementation for Java

The current implementation of Primitive Enthusiasm is for Java code, however, it can be adapted for other languages as well.

To calculate the metric, the algorithm needs to know which types should be considered primitive. We used the definition of primitive types found in the Java SE 9 language reference, Section 4.2 [7]. The following types are considered primitive: `boolean` `byte`, `short`, `int`, `long`, `char`, `float`, `double`. Additionally, we also consider the `String` as a primitive type. This is because strings in Java are immutable and could be considered as a quasi primitive type. This set of types is used as the $PrimitiveTypes$ set in the equations below.

For the calculation we used the API of OpenStaticAnalyser [1], which is an open-source, multi-language, static code analyzer developed at the Department of Software Engineering, University of Szeged.

Primitive Enthusiasm

With Primitive Enthusiasm, we limited the detection of Primitive Obsession to detecting the overuse of primitive types in method parameters. However, as it was previously stated, the need for primitive types can be application dependent, therefore we could not formulate an exact number on how much the primitive types can be used. Instead, we composed Primitive Enthusiasm as the function of the current and overall primitive type usage in the method parameter lists of a class. Our algorithm processes the examined program class-by-class and calculates the indicator for each method of the class.

Primitive Enthusiasm is a method level metric based on Formula 25 and 26. The definitions of the parameters are the following:

- $\boldsymbol{N}$ represents the number of methods in the current class.
- $\boldsymbol{M_i}$ denotes the $\boldsymbol{i}$th method of the current class.
- $\boldsymbol{M_c}$ denotes the current method under investigation in the current class.
- $\boldsymbol{P_{M_i}}$ denotes the list of types used for parameters in the the $\boldsymbol{M_i}$ method.

[1] https://github.com/sed-inf-u-szeged/OpenStaticAnalyzer

- $P_{M_i,j}$ defines the type of the jth parameter in the M_i method.

$$Primitives(M_i) = \langle P_{M_i,j} \mid 1 \leq j \leq |P_{M_i}| \wedge P_{M_i,j} \in PrimitiveTypes \rangle \tag{25}$$

Formula 25 describes how the primitive-typed parameters are collected for a given M_i method. That is, we select all those parameters from the M_i method that can be found in the previously introduced `PrimitiveTypes` set and it is returned as a list. Currently we do not care for the order of the parameters as this information is not relevant.

$$\frac{\sum_{i=1}^{N} |Primitives(M_i)|}{\sum_{i=1}^{N} \mid P_{M_i} \mid} < \frac{|Primitives(M_c)|}{\mid P_{M_c} \mid} \tag{26}$$

The left side of Formula 26 denotes the percentage of how much of the parameters of the current class are primitive types. We count the number of primitive types in the parameter list for each method in the currently processed class and divide this value with the total number of parameters in the class methods. The right side of Formula 26 calculates the percentage of the primitive types used in the currently investigated method. The number of primitive types in the current method is divided by the total number of parameters. Both sides of the inequality will calculate a number between 0.0 and 1.0 as the number of all parameters is always greater than or equal to the number of parameters that are primitive types.

If the percentage of primitive typed parameters in the current method is bigger than the percentage of primitive typed parameters in the class, the algorithm marks the method as Primitive Enthusiasm, indicating a potential place for Primitive Obsession. In case a method does not have any parameter then the calculation is skipped for that method.

It is important to note that we exclude Java setter methods, constructors, and lambda functions from the process. Setter methods are the conventional way to manipulate the primitive and non-primitive-typed class members therefore their number and their parameter type is determined by the data fields of their class. This way we eliminated reporting false positives on Java Beans. In our current approach, the methods are treated as setters if their name starts with `set`, they have only one parameter, and their return type is void. Additionally, as the metric processes method parameter types, it does not "look inside" each function, thus lambda functions are not taken into account.

Result Aggregation

The algorithm marks methods one-by-one with Primitive Enthusiasm, therefore on large projects, the list could grow ineffectively long. It can happen that it is not visible at first glance what are the most crucial places where Primitive Obsession can occur. To overcome this problem, we introduced an aggregation method that gives a class level view of the results. Thus, we rank the classes by the number of reported methods. This way we can easily see which classes might be heavily affected by Primitive Obsession.

Results

To validate usability of the proposed indicator, three projects were processed. The first project was the small GitHub project [5], which shows six variants, each with less and less primitive type parameters, as the method arguments are refactored into classes. On the first variant, our algorithm reported all five functions – the project only has five important functions – as potential methods suffering from Primitive Obsession. These five functions are indeed refactored in subsequent versions to use different types instead of primitive types.

The other two projects were the Titan [8] and ArgoUML [1] tools. For both projects there are multiple reports of Primitive Enthusiasm. Based on the initial manual validation of the results it can be stated that the class with most reports is indeed suspicious of Primitive Obsession.

Conclusion

In this work, a new indicator is presented to aid developers in detecting code elements possibly suffering from Primitive Obsession. The Primitive Enthusiasm idicator processes the method type pa-

rameters to report suspicios methods. Furthermore, an aggregation of the reported methods into a class level metric can aid the programmers to focus their attention.

We see many possibilities to improve our Primitive Obsession detection method. In the future, we would like to involve more aspects of the smell. For example, the detection of type codes, that were briefly introduced in the sample code of Section would highlight more places where this smell can occur. The usage of primitve class members should also be taken into consideration. The utilization of other metrics, like the Halstead [4] metrics which gives information about the complexity of the analysed method could also improve the reliability of our detection method.

Acknowledgment

This research was supported by the EU-funded Hungarian national grant GINOP-2.3.2-15-2016-00037 titled "Internet of Living Things".

References

[1] ArgoUML UML modeling tool (accessed: 2018-03-28). `https://github.com/stcarrez/argouml`.

[2] Martin Fowler. *Refactoring: Improving the Design of Existing Code*. Addison-Wesley, Boston, MA, USA, 1999.

[3] Aakanshi Gupta, Bharti Suri, and Sanjay Misra. A systematic literature review: Code bad smells in java source code. In *Computational Science and Its Applications – ICCSA 2017*, pages 665–682, Cham, 2017. Springer International Publishing.

[4] Maurice H. Halstead. *Elements of Software Science (Operating and Programming Systems Series)*. Elsevier Science Inc., New York, NY, USA, 1977.

[5] Refactoring from 'primitive obsession' confusion to 'tiny types' clarity (accessed: 2018-03-28). `https://github.com/stevenalowe/kata-2-tinytypes`.

[6] Satwinder Singh and Sharanpreet Kaur. A systematic literature review: Refactoring for disclosing code smells in object oriented software. *Ain Shams Engineering Journal*, 2017.

[7] The Java Language Specification, Java SE 9 Edition (accessed: 2018-03-28). `https://docs.oracle.com/javase/specs/jls/se9/jls9.pdf`.

[8] Titan distributed graph database (accessed: 2018-03-28). `https://github.com/thinkaurelius/titan`.

Multi party computation motivated by the birthday problem

Péter Hudoba, Péter Burcsi

Abstract:
The birthday problem is a widely known observation, can be found in most of the textbooks on probability. In this talk we are focusing on the following case of the problem: there are n people in a classroom and we want to decide if there are two people who were born on the same day of the year. During the decision making process, we want to keep all information secret.
We consider multiple ways to securely solve the decision problem and compare them by computational and communication aspects.

Keywords: secret multi-party computation, birthday problem, probability theory

Extended abstract

Birthday problem

One version of the birthday problem is described in [1, 2, 3] as the following: n people are selected at random. What is the probability that at least r people will have the same birthday. The widely known version is, the $n = 23$ and $r = 2$. The probability of the event that we will have two people who have been born on the same day is more than 50%.

Secure multi-party computation

Secure multi-party computation (MPC) is a field in cryptography, where the parties compute a function with their inputs jointly while keeping those inputs private. The problem in discussion, which we are focusing on is a special MPC problem. Read more in [4].

Some naive solutions

We can deduct the algorithm to the socialist millionaire problem, where we decide if the two participants have same number or not without geting any information about the other's number. If we ask all of the participants to decide if they have same birthday, we can solve the problem with $\frac{n(n-1)}{2}$ socialist millionaire solving. There are solutions for this MPC problem in [5],[6].

We can also deduct to a voting protocol. If we ask all of the participants to vote for every day, we can solve the problem in 365 voting rounds.

Security

What does "keep all information in secret" mean? Firstly, it means that at the end of the process no participant knows any other's birth date, but in this case that is also considered an information if an adversary knows that Alice and Bob have the same birthday.

This talk

In this talk we will explain multiple ways to solve the problem, show complexity and share implementations in Python.

References

[1] Frank H Mathis. A generalized birthday problem. *SIAM Review*, 33(2):265–270, 1991.

[2] David Wagner. A generalized birthday problem. In *Annual International Cryptology Conference*, pages 288–304. Springer, 2002.

[3] Morton Abramson and WOJ Moser. More birthday surprises. *The American Mathematical Monthly*, 77(8):856–858, 1970.

[4] Martin Hirt. *Multi Party Computation: Efficient Protocols, General Adversaries, and Voting*. Hartung-Gorre, 2001.

[5] Fabrice Boudot, Berry Schoenmakers, and Jacques Traore. A fair and efficient solution to the socialist millionaires problem. *Discrete Applied Mathematics*, 111(1-2):23–36, 2001.

[6] Hsiao-Ying Lin and Wen-Guey Tzeng. An efficient solution to the millionaires problem based on homomorphic encryption. In *International Conference on Applied Cryptography and Network Security*, pages 456–466. Springer, 2005.

Gaze-based Cursor Control Impairs Performance in Divided Attention

Róbert Adrian Rill, Kinga Bettina Faragó

Abstract: In this work we investigate the effects of switching from mouse cursor control to gaze-based control in a computerized divided attention task. We conducted experiments with 9 participants performing a task that requires continuous focused attention and frequent shifts of attention between the parallel tasks. Although the mouse control version was well-practiced, we hypothesized that gaze-based control would impair performance considerably because of the increased cognitive load. Our assumptions were confirmed by statistical analysis methods. The results of this study provide evidence that care should be taken in the design of interfaces controlled by human eye-gaze, which require divided attention.

Keywords: gaze-based control, eye tracking, divided attention, human performance, cognitive load

Introduction

Eye movements reveal emotions and cognitive processes, and indicate the search target during visual exploration [1]. Due to the technological advancements in eye tracking research, non-intrusive and accurate hardware solutions are readily available (see, e.g., [2]). Moreover, explosion in computer vision in recent years lead to the development of state-of-the-art appearance-based gaze estimation methods that use only images and videos from off-the-shelf monocular RGB cameras, and have an acceptable range of errors in prediction (see [3] and the references therein).

Gaze estimation has important applications in the field of human-computer interaction [4]. The traditional method for controlling the computer cursor is the PC mouse. Switching to gaze-based control has a number of advantages. Especially the performance of children with disabilities can be enhanced considerably through gaze controlled computers [5]. In the case of older adults, it may be able to compensate for the declined motor functions [6]. Gaze can also represent a superior input modality in simple computer games in terms of achievements or user experience [7].

On the other hand, when the task is cognitively demanding and requires the divided attention of users, gaze-based control might impair performance. In this study we provide evidence for this latter assumption by comparing the performance of subjects in a divided attention task between the practiced conventional mouse control and gaze-based control versions.

We designed and implemented a simplified version of the popular Train of Thought game from the Lumosity (`http://www.lumosity.com/`) online platform. Lumosity is comprised of a set of computerized games designed by scientists, each aiming to train one of five core cognitive abilities: attention, processing speed, memory, flexibility and problem solving [8]. We will refer to our version of the Train of Thought game simply as the *Train Game*. We do not detail the design and implementation process due to space limitations, but only describe briefly the purpose of the task.

The Train Game tests players divided attention and working memory by requiring them to continuously focus on multiple simultaneous targets, to switch frequently between them keeping track of each one. The task of the user is to direct continuously oncoming trains to their color-matching destination through flipping switches at forks with mouse clicks and changing this way the direction of the tracks and the path of the moving trains.

The paper is organized as follows. In Section we describe the experiments with the Train Game and differentiate two types of user errors. Section presents the results of the statistical analysis. This is followed by a discussion in Section . Finally, Section concludes the paper and mentions future directions.

Methods

Participants and experiments

We have performed a series of experiments with 10 participants, who were asked to play with the mouse control and the gaze-based control of the Train Game. The volunteers were aged between 25 and

30 (mean age was 27 years, SD=1.76), had normal or corrected-to-normal vision and reported no attention disorders nor color vision deficiency. Also they were naive in the sense that they were not familiar with the Train Game as well as did not have prior experience in Lumosity games. The participants were instructed that data about their gameplays will be logged for further analysis and they were asked to sign a consent form before the experiments.

The experiments with the mouse control version of the Train Game lasted several days, with multiple trials played each day. We manipulated the difficulty of the game according to the score of the players from the previous trial. Based on this, the experiments were separated into three phases: beginner, intermediate and advanced.

For the experiments with the gaze control version of the Train Game, 9 out of the 10 participants were invited back for ten additional trials. For gaze tracking the Tobii EyeX Controller (https://tobiigaming.com/product/tobii-eyex/) device was used, which is attached to the bottom of the monitor, has a sampling rate of 60 Hz and requires personal calibration before each data collecting session. The traditional mouse cursor was replaced by a yellow dot displayed at the screen coordinates of the gaze direction. To reduce the noise of the gaze tracker device, we applied a moving average window on each of 5 consecutive samples. If the distance of consecutive gaze points and the center of a switch node was below a threshold for longer than 500 ms, a mouse click was simulated that results in flipping the switch.

User error types

The details of the experiments with the mouse control version of the Train Game are presented in our previous work [9]. For the purposes of this study we selected 10 consecutive trials from the intermediate phase to compare them with the 10 gaze control trials.

The performance of the participants is determined by the user errors, which can be separated into two categories. (i) Errors of omission are the cases when the player misses an action, are the more common ones and can have several causes: the place of the action is outside the visual field, too little time to perform multiple parallel tasks. (ii) Errors of commission occur when the player performs a prohibited action, and do not correct it. These latter actions are the more rare ones in the Train Game and happen when the player confuses two colors, performs an action too early or acts recklessly because of pressure.

In our analysis we computed the number of each type of user errors and compared the means between the mouse control and the gaze-based control versions using the repeated measures analysis of variance (ANOVA) statistical model.

Results

We calculated the total number of user errors for each trial in both conditions (mouse control and gaze control). There was a statistically significant difference in means of user errors between the two control types, as determined by the repeated measures ANOVA, $F(1,8) = 61.19$, $p < 0.001$. Figure 1 shows the mean of the user error numbers across trials in each of the two control versions of the Train Game, separately for every subject.

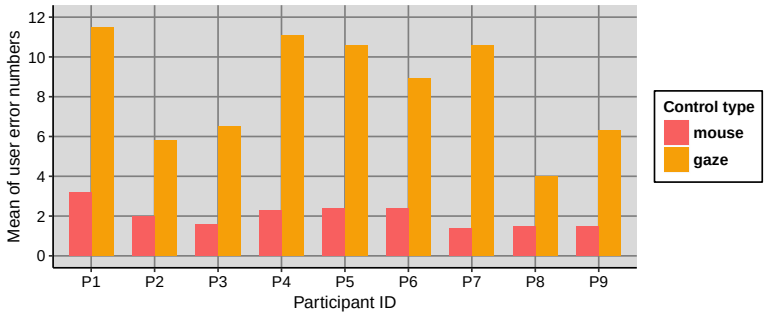


Figure 1: Comparison of mean of overall user error numbers separately for participants.

The proportion of the commission type of errors to the total number of errors was also calculated. Figure 2 compares these percentages computed over all 10 trials for each participant. We can see that generally the proportion of commission errors is considerably higher in the gaze version. Also, the repeated measures ANOVA for proportion of commission errors (computed for each trial separately) showed significant main effects of control type (mouse vs. gaze), $F(1, 8) = 20.28$, $p = 0.002$.

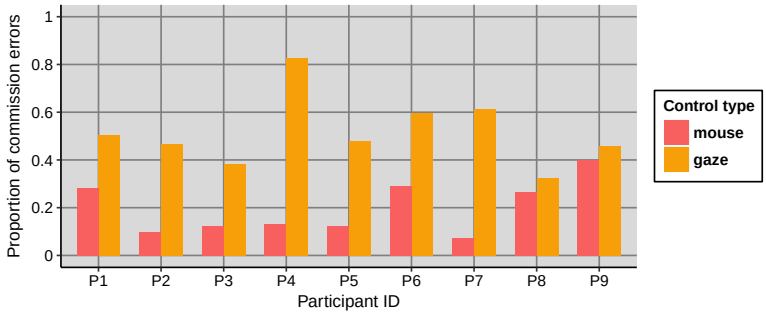


Figure 2: Comparison of proportion of commission type user errors separately for participants.

Discussion

We tested the effect of switching from mouse control to gaze-based control in a complex divided attention task. The number of user errors was increased in the latter version as seen on Figure 1, and this change was also statistically significant, as determined by the repeated measures ANOVA. One might argue that our results are biased because of the latency for mouse click simulation or the selection of the trials for comparison. The latency contributes indeed to cognitive load because it limits the exploration time. However, the intermediate phase was the part where subjects were familiar with the Train Game and could play comfortably after the beginner phase. In the advanced phase difficulty was high in order to test the effects of time pressure. More importantly, we also compared the proportion of errors of commission, shown on Figure 2, and have seen that this also increased significantly in the gaze control version. Consequently, performance decrements can be attributed to reckless actions from the increased cognitive load and not time constraints.

The sample size in this study is small, which limits the generalizability of the results. Also,the performance differences are partially due to the lack of practice with gaze-based control. However, not only was the overall number of errors larger, but the proportion of errors of commission increased consistently for all of the subjects in our experiments. This is a result of increased cognitive load. Furthermore, our conclusions are in line with studies investigating gaze-based control in demanding tasks. For example, Smith and Graham [10] found that mouse control was easier in a first-person shooter game. This may be related to the "Midas Touch" problem, which could be expanded in our case too.

Conclusion and future work

Eye movements are an important cue for inferring human behaviour. Gaze can indicate emotions, intentions, objects of interest and focus of attention. Gaze estimation has relevant applications in the design of interfaces that facilitate human-computer interaction. Gaze control had a negative impact on the performance of users in our experiments. These results provide evidence that in the case of complex tasks requiring the divided attention of users, switching from the traditional mouse control to gaze-based control may not be the best choice. Our efforts suggest that at least considerable amount of testing and/or training is necessary before using such interfaces in real-life scenarios.

The combination of mouse and gaze-based input may represent a suitable solution in such cases. Future studies should also investigate the effects of gaze control on the strategies of subjects, which influence performance in multitasking to a great extent.

Acknowledgement

The research has been supported by the European Union, co-financed by the European Social Fund (EFOP-3.6.2-16-2017-00013).

References

[1] Ali Borji, Andreas Lennartz, and Marc Pomplun. What do eyes reveal about the mind?: Algorithmic inference of search targets from fixations. *Neurocomputing*, 149:788 – 799, 2015.

[2] Agostino Gibaldi, Mauricio Vanegas, Peter J. Bex, and Guido Maiello. Evaluation of the Tobii EyeX Eye tracking controller and Matlab toolkit for research. *Behavior Research Methods*, 49(3):923–946, 2017.

[3] Xucong Zhang, Yusuke Sugano, Mario Fritz, and Andreas Bulling. MPIIGaze: Real-world dataset and deep appearance-based gaze estimation. *CoRR*, abs/1711.09017, 2017.

[4] Päivi Majaranta and Andreas Bulling. *Eye Tracking and Eye-Based Human-Computer Interaction*, pages 39–65. Springer London, London, 2014.

[5] Eva Holmqvist, Sandra Derbring, and Sofia Wallin. Participation through gaze controlled computer for children with severe multiple disabilities. *Studies in Health Technology and Informatics*, 242:1103–1108, 2017.

[6] Atsuo Murata. Eye-gaze input versus mouse: Cursor control as a function of age. *International Journal of Human-Computer Interaction*, 21(1):1–14, 2006.

[7] Roman Bednarik, Tersia Gowases, and Markku Tukiainen. Gaze interaction enhances problem solving: Effects of dwell-time based, gaze-augmented, and mouse interaction on problem-solving strategies and user experience. *Journal of Eye Movement Research*, 3(1), 2009.

[8] Joseph L. Hardy, Faraz Farzin, and Michael Scanlon. The science behind lumosity, Version 2, 2013. Lumos Labs, Inc.

[9] Róbert Adrian Rill, Kinga Bettina Faragó, and András Lőrincz. Strategic predictors of performance in a divided attention task. *PLOS ONE*, 13(4):1–27, 04 2018.

[10] J. David Smith and T. C. Nicholas Graham. Use of eye movements for video game control. In *Proceedings of the 2006 ACM SIGCHI International Conference on Advances in Computer Entertainment Technology*, ACE '06, New York, NY, USA, 2006. ACM.

Towards a Classification to Facilitate the Design of Domain-Specific Visual Languages

Sándor Bácsi, Gergely Mezei

Abstract: Domain-specific visual languages (DSVLs) are specialized modeling languages that allow the effective management of the behavior and the structure of software programs and systems in a specific domain. Each DSVL has its specific structural and graphical characteristics depending on the problem domain. In the recent decade, a wide range of tools and methodologies have been introduced to support the design of DSVLs for various domains, therefore it can be a challenging task to choose the most appropriate techniques for the design process. Our research aims to present a classification to guide the identification of the most relevant and appropriate methodologies in the given scenario. The classification is capable enough to provide a clear and precise understanding of the main aspects that can facilitate the design of DSVLs.

Keywords: domain-specific visual languages, modeling, classification

Introduction

The graphical interface of domains-specific visual languages (DSVLs) provides an improved abstraction of the problem that allows the rapid development for a specific domain. Expressiveness, flexibility and usability are the most important aspects in the effective development with a DSVL. Each problem domain requires a different visual representation to meet the requirements of the targeted domain, thus it is essential to choose the most appropriate representational concepts in design-time. The exactness of the choice depends on how expressively the chosen concepts describes the DSVL and on the specific needs of the targeted domain. To support the design process, various kinds of classifications have been created in the last decades. There are classifications mainly based on formalism [1], while other classifications are based on metamodeling approaches [4]. On the other hand, there are existing classification frameworks which support the design of visual languages [2]. Due to the increasing use of DSVLs, a wide range of new tools and methodologies have been introduced recently based on completely new ideas. Our research aims to analyze and compare the most relevant methodologies on a larger scope which can support the design of new domain-specific visual languages.

In this paper, we present the main results of our classification methodology. The classification is mainly based on the nature of the graphical objects that compose the visual language, the connection types among the graphical objects and the composition rules. In our research, we have analyzed a wide range of existing DSVL methodologies and also created several case studies for different domains to exemplify the most relevant graphical and structural characteristic. We have used two metamodeling frameworks (Eclipse Modelling Framework [9] and Visual Modeling and Transformation System [8]) and a visual programming editor builder (Google Blockly [7]) to examine the most applicable methodologies. In this paper, we provide a summary of our work focusing on the results rather than on the case studies applied.

Towards the Classification of Visual Languages

In this section, we present the main aspects of the classification. Each subsection of this section represents one dimension of the classification.

Methods of the abstract syntax definition

There are two key methods for the abstract syntax definition of a DSVL: metamodeling methodologies and non-metamodeling approaches.

Metamodeling methodologies provide possibilities for the definition and management of DSVLs based on the abstract notion of visual entities and of relations among them. These frameworks are capable of specifying the abstract syntax of a DSVL and expressing the additional semantics of existing information. The metamodel can expressively define the structure, semantics, and constraints for a

family of graphical models. Hence, complex structures and relations can be flexibly described by the usage of metamodeling concepts.

While metamodeling methodologies are based on various kinds of instantiation techniques, non-metamodeling approaches provide a template-based structure for creating visual entities. The main characteristic of these approaches is that they have a limited set of features which can be used on the different abstraction levels, thus complex structures cannot be visualized flexibly and expressively. One of the newest non-metamodeling approaches is Blockly [7]. It supports a large set of features for different domains. In Blockly the graphical objects are called blocks which can be customized as the basic building elements of the language. Due to the template-based and weakly typed structure of Blockly, type constraints cannot be applied.

Relation type

Based on the relation type, domain-specific visual languages can be grouped into two subclasses: containment-based and connection-based subclasses.

In containment-based languages, the graphical entities can be embedded in each other to express visual sentences of the targeted domain, therefore, no connections are needed. Beside embedding, the graphical entities can be attached to other entities (as a method and its parameters) and chained together (as in a call stack). A pre-defined set of containment rules or constraints have to be defined to restrict the way of embedding, attaching and chaining. Blockly and Scratch [6] are widely used examples of containment-based languages. Both Blockly and Scratch support building blocks that can be connected like puzzle pieces in order to create visual sentences.

Connection-based languages consist of two different kinds of building elements: entities and connections, i.e. nodes and edges. While the data is basically expressed by entities, the flow of the model and the relations among entities are defined by connections. Connections may also have properties to ensure the customization of the relations among entities. Beside the connection-based patterns, entities may have containment-based nature, for example, they can be embedded in each other.

Flow type

Based on the flow type, domain-specific visual languages can be grouped into three subclasses: data flow languages, control flow languages and languages with no flow.

Data flow visual languages visualize the flow of data focusing on the steps of data processing. Data flow concepts are based on the idea of disconnecting computational actors into stages that can execute concurrently. Data flow DSVLs visualize the processes that are undertaken, the data produced and consumed by each process, and the storing graphical objects needed to hold the data. It is possible to visualize what the system will accomplish by the flow of data.

Control flow visual languages visualize the logic of computation by describing its control flow. Control flow DSVLs graphically express the order in which instructions or statements are executed or evaluated. The graphical objects mainly represent the control structures and conditional expressions of the language, thus it is possible to visualize how the system will operate by the flow of control.

There are DSVLs which are neither data flow nor control flow because they target a static domain problem. These languages can visually represent the structure of a system or a program, therefore no flow has to be described. A widely used example of no-flow graphical modeling languages is the UML class diagram, in which the structure of the system is described by the classes and the connections among them.

The way of the problem description

Based on the way of the problem description, domain-specific visual languages can be grouped into two subclasses: imperative and declarative languages.

Declarative visual languages describe the logic of computation. For example, SparqlBlocks [5] is a declarative DSVL developed in Blockly. Declarative languages visualize sets of declarations or declarative statements. Each of these visual declarations has a meaning depending on the targeted domain and may be understood independently. A declarative style of visualization helps to understand the problems of the targeted domain and the approach that the system takes towards the solution of the problem, but is less expressive on the matter of mechanics which describe the flow of the system.

Imperative visual languages consist of visual statements that change the state of a program or a system. For example, Scratch is an imperative visual programming language. The visualized statements express the way of execution of which results in a decision being made as to which of two or more visualized paths to follow. In imperative languages, the visual sentences can be created by sequences of commands, each of which performs some action. These actions may or may not have a dedicated meaning in the targeted problem domain.

Visual representation

DSVLs have a visual concrete syntax used for the representation of graphical elements and connections. Based on the visual representation, there are two key design aspects: iconic and diagrammatic visual representation.

Entities are visualized by icons in iconic languages. For example, Lego Wedo 2.0 Software [13] provides an iconic visual language for educational purposes. The iconic language is a structured set of the related icons. An icon can be composed of other icons or can be attached to another one, thus expressing a more complex visual concept. Some icons can be immanently deceptive, some can only be interpreted within a certain domain context, thus iconic visual languages may have their disadvantages.

Diagrammatic languages mainly composed of elements with a pre-defined symbolic representation of information. The building blocks of diagrammatic languages such as geometric shapes are often connected by lines, arrows, or other visual links. Chart-like, schematic-like and graph-based visual languages are the most widely used examples.

Conclusions

In this paper we have presented several aspects of the classification for domain-specific visual languages. We believe that this classification can be used as a guide while designing DSVLs. Hence, with the help of these guidelines it is now possible to analyze the characteristics of the language, and to associate it to an appropriate class or dimension.

On the other hand, we have analyzed the features of EMF, VMTS and Blockly based on different case studies that we have created for our classification methodology. We realized that due to the limitations of Blockly, many complex problems cannot be described expressively because aggregations, references and composition rules are missing from its developer framework. Despite the limitations of Blockly, it provides a flexible and easy way to learn to design imperative and control flow DSVLs based on containment-based aspects. Unlike Blockly, both EMF and VMTS provide a large feature set for the abstract syntax definition, but they are not as effective as Blockly in definition of containment-based languages. Based on EMF, Sirius [10] provides useful features for the customization of concrete syntax.

In the future, we aim to create a framework to support the design of visual-domain specific languages based on a questionnaire built upon the classification presented. We plan to create a framework to support an intuitively usable way of designing DSVLs even for complex language constructs. We are also working on new case studies and analyzing other existing approaches to create a more detailed classification.

Acknowledgements

The research has been supported by the European Union, co-financed by the European Social Fund (EFOP-3.6.2-16-2017-00013).

References

[1] K. Marriott and B. Meyer. On the classification of visual languages by grammar hierarchies. Journal of Visual Languages and Computing, 8(4):375–402, 1997.

[2] Costagliola, G., A. Delucia, S. Orefice, and G. Polese, A Classification Framework to Support the Design of Visual Languages. Journal of Visual Languages and Computing, 2002. 13: p. 573-600.

[3] J. Sprinkle and G. Karsai, "A Domain-Specific Visual Language For Domain Model Evolution," Journal of Visual Languages & Computing, vol. 15, no. 3, pp. 291-307, 2004

[4] P. Bottoni and A. Grau, "A suite of metamodels as a basis for a classification of visual languages," in Visual Languages and Human Centric Computing, 2004 IEEE Symposium on, sept. 2004, pp. 83-90.

[5] Bottoni, P., Ceriani, M.: SPARQL playground: A block programming tool to experiment with SPARQL. In: Proceedings of the International Workshop on Visualizations and User Interfaces for Ontologies and Linked Data (VOILA@ISWC 2015).

[6] Scratch. Retrieved March 4, 2018, from https://scratch.mit.edu/

[7] Blockly Website. Retrieved March 8, 2018, from https://developers.google.com/blockly/

[8] VMTS Website. Retrieved March 10, 2018, from
https://www.aut.bme.hu/Pages/Research/VMTS/Introduction

[9] EMF. Retrieved March 16, 2018, from https://www.eclipse.org/modeling/emf/

[10] Sirius. Retrieved March 18, 2018, from https://www.eclipse.org/sirius/

[11] Ulrik Pagh Schultz. Graphical/Visual DSL. Retrieved March 23, 2018, from
http://websrv0a.sdu.dk/ups/SSE02/slides/lecture-10.pdf

[12] Iconic Visual Languages. Retrieved March 23, 2018, from
https://people.cs.pitt.edu/ chang/365/sk1.html

[13] Lego Wedo 2.0 Software. Retrieved March 30, 2018, from
https://education.lego.com/en-us/downloads/wedo-2/software

Use data mining methods in quality measurement in the education systems

Sándor Balázs Domonkos, Németh Tamás

Abstract: Our basic problem is rooted in the education systems, where they want to measure and evaluate the pedagogical work from year to year and watch for pedagogical developments. These measures can be used for the individuals to get informations on which field they need to improve and can be used for rewarding systems. These measurements can be achieved by different methods, in this case our method is the surveying in the classes,direct approach person to person surveys and demand and satisfaction measurements for every person. These surveys are precomposed with discussions about the needed attributes. We get real datas from 58 different schools from 2007-2016 , nearly 8000 educators. All these surveys-ratings and other datas get collected and processed to get in a usable form. The schools make a statistics with the collected datas every year, but the statistics has really person dependents and because of that these has a lots of distorsions. For example a human focused teacher can give bad points for a informatics teacher class, because they dont share a territorial interest.To get a workaround for these statistics "personal" dependencies, we use the pagerank algorithm with the Comparability graph[1]. With the comparability graph we could compare two attributes with each other not heavily depended on the people who fill the survey and we could make a new graph for every attributes. After that on those graphs we can use the pagerank algorithms to get the datas out that we want to examine and get further consequences. For these datas we should tell they are lead to a much more usable development curves about the educators qualities. Important thing is these datas dont suffer the distorsions of the statistic ones has.

Keywords: data mining, pagerank, comparability graph

Introduction

The main problem is raising from the education system qualilty and valuation system about the educators. This system is really important for the schools because only this way they can monitor the educators and other workers qualities and working capabilities. These datas that they collected via surveys are not just used to person evaluation and ratings, but for to get a view about the workload for each individual, so it can help for example to make a better timetable for everyone with this datas too. For bigger view we can get more datas about the leadership and the lecturer and parents connections too and the way they communicate with each other. But this article is focusing on the quality measurement on the lecturers, with our method they get more accurate data. So they can get well earned rewards and get a better feedback on which area they are on a good level and where they need to improve. So we examined the datas and the statistical style old system and we get the conclusion that the statistical based system is can be very noisy and heavily depends on the person to person connections and sometimes the given teacher connection to another subject. For example literature teacher can give only bad points for informatics and math teachers and good points only for literature teachers, because they dont share a territorial interest or they are not in a good connection in the workplace or any other issues they can have. So statistics methods cannot overcome this types of noise and predilection. Because of this the statistics method have a very high dispersion on the results and for year to year compliment the fluctuation is very heavy on this results.

Graph Based Solution

Our solution for this problem is based on graphs. We analize the raw datas try out different methods to break up the datas for different viewpoints and build different types of graphs. After a lots of experience we decide to use Comparability graphs for every attributes. First of all we need to break up the datas for 3 parts: evaluating lecturer, attributes for the evaluated lecturer, evaluated lecturer. So in this way we can construct comparability graphs for every attributes. In this way 2 evalutaion can be compared if they are from the same evaluator lecturer and for the same attributes, in this way the data noise is dampening down,because its compare data from the same evaluator. From these datas we construct

Comparability graphs for every attributes. On these new graphs we can use the PageRank algorithm to get quantified results for every evaluated lecturer for every attributes without the distorsion of the personal conflicts or any other type of distorsions mentioned above. The pagerank algorith give a more data about the evaluating lecutrers too for each attributes and after the pagerank is done these values are just more well adjusted for every attributes for ever evaluated lecturer. For example on a statistics old styled method if someone is get a few bad results from personal conflicts, these bad ratings can really pull down the person numbers , but with our method the comparability graph for attritbutes processed with pagerank methods these bad ratings can ease down the datas, cause the algoritm will ease out if someone give only bad points for an individual and good points for everyone else or give bad points for everyone and this can be takint into account with a less impact, and these are all for the good cases too, so not just only the bad ratings.

Results

The quantified values are less likely to depend on the evaluator person and much more like depends on evaluated person. This method the year to year improvements are take shape in a better way and more readable way. This way we not just only get the results for the evaluated people attributes it can work backway too, we can get datas from the evaluators too, so we can make some conclusion in the lecturers have a personal or work related problems with each other. For results we get a much lower dispersions level and with a lower dispersion and very low fluctuation level we can much more easier fit line to yearly improvement attributes too. The average dispersion level for the results can be 30-50 percent better than the statistics ones.

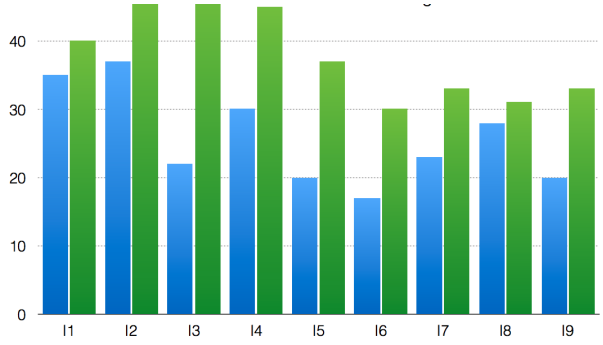


Figure 1: Blue pagerank, green statistical

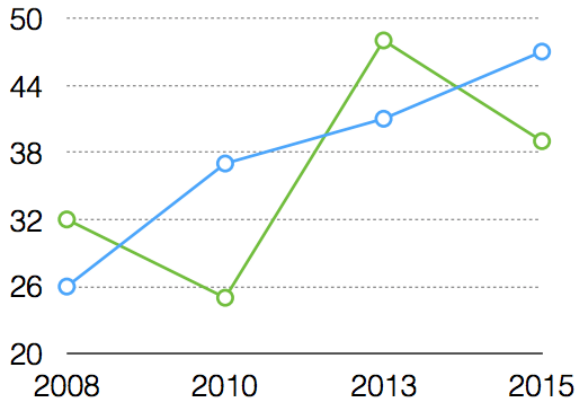


Figure 2: Blue pagerank, green statistical

Acknowledgements

This work was supported by Enaplo and snw systems.

References

[1] A. London. A local PageRank algorithm for evaluating the importance of scientific articles , *Annales Mathematicae et Informaticae* 44:131-140 2015.

Measuring the similarity of two cohorts in the n-dimensional space

Szabolcs Szekér, Ágnes Vathy-Fogarassy

Abstract: Measuring the similarity of the case and control group in clinical studies has always been an important but also difficult task. Several statistics-based methods aimed at this exist but most of them utilize dimension reduction or estimation, therefore, there are certain cases where they are not adequate. In this paper, we propose 3 dissimilarity-based measures capable of evaluating case-control group pairs without the loss of valuable information.

Keywords: similarity measure, control group, case group, non-paired matching, paired matching

Introduction

Observational studies in the field of social sciences, natural sciences and engineering are often based on comparative analysis. In these cohort-based studies, people are classified into two independent groups (case and control groups), and the conclusion is drawn on the basis of the similarities and differences between the two groups. Although the comparison-based scientific analyses significantly differ in the applied methodology and study design principles [1], due to their comparative nature they have strong scientific evidence [2]. Generally, the selection of the case group can be carried out based on the study aims, but the determination of the control group has difficulties and it raises many questions [3, 4]. Various methods have been proposed for the selection of the control group. Some methods utilize randomized sampling or stratified sampling [5, 6], while others are based on propensity score matching (PSM) [7] or on the selection of the k-nearest neighbors [8, 9]. All these methods try to form a control group which contains individuals that are similar to the individuals of the case group in certain characteristics, and in specific property or properties, they differ from them. But the measurement of the similarity of the case and control groups is not a trivial task. In this paper, we give a short overview of the applied similarity measuring approaches and propose three similarity measures to evaluate the degree of similarity of equally sized case and control groups in the n-dimensional space.

The rest of the paper is organized as follows. Section 2 briefly presents the basic approaches applied in this field and introduces our proposed measures. In Section 3, some comparative results are briefly presented. Finally, conclusions are drawn in Section 4.

Evaluation of the similarity of the case and control groups

Measuring the similarity of groups used in comparative analyses is an important task. The evaluation can happen by measuring the similarity of paired individuals from the case and control groups (paired evaluation) or by assessing the similarity of the case and control groups (non-paired evaluation). Most of the applied non-paired methods are Goodness of Fit (GoF) tests (e.g. Kolgomorov-Smirnov test, Bhattacharyya distance, Matusita distance) [10] evaluating the distribution of the two groups. Using a GoF test, it is possible to evaluate a one-dimensional distribution (that is the similarity of a certain property), but it is nearly impossible for higher dimensions [11]. However, people as the elements of the groups are characterized not by one but by many features. On the other hand, if the elements of the control group are selected by propensity score matching, the similarity of the case and control elements is measured again only in one dimension, namely as the dissimilarities of the propensity scores. As the propensity score is an estimated value, the similarity measurement is made in a lossy compressed 1-dimensional space, and not in the original feature space of the elements. Contrary to these methods, our aim is to measure the similarity of the case and control groups in the original high-dimensional feature space of the individuals. For this reason, we propose 3 dissimilarity measures: 2 for paired and 1 for non-paired evaluation. These measures can be considered as a normalized average of dissimilarities, but they differ in the interpretation of the term dissimilarity.

Paired evaluation

Nearest Neighbor Index

The first measure is called Nearest Neighbor Index (NNI) and it is quite strict. NNI checks for each

attribute whether the case-control entity pairs are the closest neighbors to each other on that attribute. Pair elements are determined by the applied matching method. For categorical features the dissimilarity is 0 if the values of the attributes of the individuals are identical, otherwise 1. For continuous attributes, the dissimilarity is 0 if and only if the sample-control pair is the closest to each other pursuant to the examined attribute. Finally, the NNI is calculated as the average of the dissimilarities calculated in each dimension.

Dissimilarity for categorical features:

$$d_{ij}^k = \begin{cases} 0 & if \quad a_{ik} = b_{jk} \\ 1 & if \quad a_{ik} \neq b_{jk} \end{cases} \tag{27}$$

Dissimilarity for continous features:

$$d_{ij}^k = \begin{cases} 0 & if \quad \left|a_{ik} - b_{jk}\right| = min\left(\left|a_{ik} - b_{lk}\right|\right), \quad l = 1, 2, ..., N \\ 1 & if \quad \left|a_{ik} - b_{jk}\right| > min\left(\left|a_{ik} - b_{lk}\right|\right), \quad l = 1, 2, ..., N \end{cases} \tag{28}$$

Nearest Neighbor Index:

$$NNI(A, B) = \frac{\sum_{(a_i, b_j) \in P} \sum_{k=1}^{n} d_{ij}^k}{nN} \tag{29}$$

where a_i is an element of the case group ($a_i \in A$), b_j is an element of the control group ($b_j \in B$), $(a_i, b_j) \in P$ yields that they are matched case-control pairs, a_{ik} yields the value of the k-th dimension of a_i, b_{jk} analogously for the b_j element, n is the number of the characterizing features and N yields the number of individuals in either of the groups.

Global Dissimilarity Index

It is apparent that NNI checks for every dimension if the case-control pairs are closest to each other in that dimension, however, it does not consider the distance between them. The Global Dissimilarity Index (GDI) is a paired measure that is meant to account for this weakness.

GDI measures the dissimilarity for nominal features as the function of the number of different values, for ordinal features as the difference of ranks and for continuous features as the normalized distance. The calculation of GDI happens analogously as for NNI (Equation 29). The statement about paired elements still holds.

Dissimilarity for nominal features:

$$d_{ij}^k = \begin{cases} 0 & if \quad a_{ik} = b_{jk} \\ \frac{1}{M_k} & if \quad a_{ik} \neq b_{jk} \end{cases} \tag{30}$$

Dissimilarity for ordinal features:

$$d_{ij}^k = \begin{cases} 0 & if \quad a_{ik} = b_{jk} \\ \frac{\left|r_{a_{ik}} - r_{b_{jk}}\right|}{M_k - 1} & if \quad a_{ik} \neq b_{jk} \end{cases} \tag{31}$$

Dissimilarity for continuous features:

$$d_{ij}^k = \frac{\left|a_{ik} - b_{jk}\right|}{max\left(s_{lk}\right) - min\left(s_{lk}\right)}, \quad s \in \{A \cup B\}, \quad l = 1, ..., 2N \tag{32}$$

where M_k is the number of possible values along the k-th dimension, r yields the ordered rank for the ordinal attributes, $min\left(s_{lk}\right)$ is the minimal value and $max\left(s_{lk}\right)$ is the maximal value along the k-th dimension.

Non-paired evaluation

The above-mentioned methods measure the dissimilarity by determining the pairwise dissimilarities for each case-control pair. It is possible that only the similarity of the distributions of the characterizing

features counts and the pairwise matching of the individuals is not relevant. On that score, we implemented a distribution-based measure called **Distribution Dissimilarity Index** (DDI). DDI is based on the histogram disparities of the case and control groups in each dimension. This method relies on the absolute deviation of the frequency of each property value relative to the size of the control group and the number of characterizing features. If the individuals are characterized by continuous values, the values have to be discretized before the calculation of the frequency values of the histogram.

$$DDI(A,B) = \frac{\sum_k \sum_{v \in V_k} \left| f_{kv}^A - f_{kv}^B \right|}{nN}, \quad k = 1, ..., n \tag{33}$$

where f_{kv}^A yields the frequency of the v-th value in the k-th dimension in the case group, f_{kv}^B analogously for the control group, V_k is the number of possible values along the k-th dimension.

Results of a short case study

To demonstrate the previously presented measures, we generated a benchmark dataset containing 1000 elements characterized by 8 variables (2 binary, 2 ordinal, 1 nominal and 3 continuous): 1 Bernoulli random variable with a probability value of 0.5, 1 Bernoulli random variable with a probability value of 0.3, 1 binomial variable with 3 trials and a probability value of 0.5, 1 uniform discrete variable in the range of $[0, 5)$, a uniform discrete variable in the range of $[0, 4)$, 1 uniform variable in the range of $[0, 2)$, 1 variable with normal distribution with a mean of 2 and standard deviation of 0.5 and 1 variable with normal distribution with a mean of 1 and standard deviation of 2. Additionally, a portion of the generated dataset was distorted with noise: $1\,\%$, $5\,\%$, $10\,\%$, $25\,\%$, $50\,\%$, $75\,\%$, $90\,\%$ and finally $100\,\%$ of the dataset was distorted along each dimension. In total, 9 case-control group pairs comprised our test scenario, in which dissimilarities of all pairs were evaluated by NNI, GDI, and DDI. The result of the evaluation can be seen in Table 1.

Table 1: DDI, NNI and GDI values for the 9 case-control group pairs. The presented values are dissimilarities in the range of $[0, 1]$. The smaller the value, the more similar the given case and control groups are.

Noise	$0\,\%$	$1\,\%$	$5\,\%$	$10\,\%$	$25\,\%$	$50\,\%$	$75\,\%$	$90\,\%$	$100\,\%$
NNI	0.000	0.009	0.043	0.085	0.214	0.426	0.645	0.772	0.851
GDI	0.000	0.004	0.019	0.038	0.096	0.192	0.291	0.347	0.385
DDI	0.000	0.003	0.008	0.015	0.036	0.054	0.080	0.091	0.097

As previously mentioned, NNI is the strictest measure, so it is especially sensitive to noise and dissimilar data. The 0.851 dissimilarity value is a proof of that behaviour. It is important to mention, that total dissimilarity (when the dissimilarity value is 1) is only achievable in extreme cases. These extreme cases are where the compaired values are at the opposite ends of the range of the examined variable. The statement about strict nature also holds for GDI, while DDI, the non-paired measure is noticably less sensitive, reaching only 0.097 dissimilarity when the whole dataset is distorted.

Conclusion

Control group evaluation is a non-trivial task and the selected similarity measure greatly affects the evaluation of research results. In this paper, we proposed 3 measures capable of evaluating the similarity of case and control groups as n-dimensional data in contrast to traditional methods that apply dimension reduction or estimation. All measures serve different purposes: DDI is recommended for non-paired evaluation, while NNI and GDI are recommended for paired evaluation. While in the course of scientific research analysts tend to consider only one criteria to evaluate the similarity of case and control groups, this article points to the fact that it is worth considering several aspects together.

Acknowledgements

We acknowledge the financial support of Széchenyi 2020 under the EFOP-3.6.1-16-2016-00015.

References

[1] J. W. Song and K. C. Chung, "Observational studies: Cohort and case-control studies," *Plastic and Reconstructive Surgery*, vol. 126, no. 6, p. 2234–2242, 2010.

[2] B. Everitt and C. R. Palmer, *Encyclopaedic companion to medical statistics*. Wiley, 2011.

[3] S. Wacholder, D. T. Silverman, J. K. Mclaughlin, and J. S. Mandel, "Selection of controls in case-control studies: Ii. types of controls," *American Journal of Epidemiology*, vol. 135, p. 1029–1041, Jan 1992.

[4] N. S. Weiss and T. D. Koepsell, "Epidemiologic methods," 2014.

[5] N. P. Jewell, "Least squares regression with data arising from stratified samples of the dependent variable," *Biometrika*, vol. 72, no. 1, p. 11, 1985.

[6] S. Singh, "Stratified and post-stratified sampling," *Advanced Sampling Theory with Applications*, p. 649–764, 2003.

[7] P. C. Austin, "An introduction to propensity score methods for reducing the effects of confounding in observational studies," *Multivariate Behavioral Research*, vol. 46, no. 3, p. 399–424, 2011.

[8] S. Szekér and A. Vathy-Fogarassy, "Kontrollcsoport-generálási lehetőségek retrospektív egészségügyi vizsgálatokhoz," in *Neumann Kollokvium konferenciakiadványa, Orvosi Informatika*, p. 146, 2016.

[9] S. Szekér and A. Vathy-Fogarassy, "Novel k nearest neighbor-based control group selection methods," in *13th Miklós Iványi International PhD and DLA Symposium - Abstract Book: Architectural, Engineering and Information Sciences*, p. 124, Pollack Press, 2017.

[10] P. W. Mielke and K. J. Berry, *Permutation Methods*. Springer, 2007.

[11] G. Fasano and A. Franceschini, "A multidimensional version of the kolmogorov–smirnov test," *Monthly Notices of the Royal Astronomical Society*, vol. 225, no. 1, p. 155–170, 1987.

Should we omit the practical aspects in modeling the server clusters?

Thanh-Binh V. Lam

Abstract: This paper investigates the impact of some practical aspects which are simplified on modeling the server clusters. Three considered aspects are S1) the distribution of the switching times, S2) the avoidance of turning off the servers in setup process, and S3) the presence of the shutdown time.

The results demonstrate that the accuracy of the calculation for energy consumption metrics in the server clusters might be improved unless we omit these practical aspects. In particular, both S2 and S3 expose considerable impact on the accuracy of the calculation for the average energy consumption. Taking these aspects into account might support the operators in planning and calculating the energy consumption for their server clusters.

Keywords: Server clusters, setup time, shutdown time, analytical model, practical aspects.

Introduction

The accuracy of the calculation for performance measures and energy consumption metrics is one of the important factors in managing server clusters of the IT service operators. Precisely, it helps them in planning as well as controlling the operation of their system. When modeling server clusters, however, the authors usually omit some practical aspects in the operation of servers, such as, setup time, shutdown time, procedures to turn on and turn off a server, etc. As a consequence, the operation of the server cluster does not behave as its natural dynamics. Throughout this paper, we demonstrate that these practical aspects play an important role in improving the accuracy of the calculation for performance measures and energy consumption metrics in the server clusters.

The authors [1, 2, 3, 4, 5] omitted the shutdown phase of a server which is impractical. More precisely, they took the setup time into account only. They assumed that a server can be turned off immediately without any cost in terms of energy consumption. Particularly, the authors [1] proposed to stop servers in setup process. Moreover, the authors [5, 6] assumed that a block of servers can be powered up or powered down simultaneously. From a practical point of views, even the servers are homogeneous and the commands to power them up or down are initiated at the same time, they could not finish their setup/shutdown periods in the same seconds. The authors [7] first took into account these aspects within some different workloads. However, they did not consider the size of server clusters. The question is whether these aspects impact on the accuracy of the calculation for performance measures and energy consumption metrics in server clusters in the case of varying the cluster size?

More precisely, our goals in this paper are

- to study the impact of distribution of switching times to the accuracy of the calculation for performance measures,
- to check whether turning off a server in setup process effects to the calculation for energy consumption in the system,
- to investigate the presence of shutdown time in the accuracy of the calculation for performance measures,
- to review the impact of cluster size on the calculation for performance parameters in the server clusters.

The rest of the paper is organized as follows. Section presents the abstract model which are considered throughout this paper. The simulation experiments are demonstrated in Section . Finally, Section concludes our work.

System modeling

We follow [7] to apply the abstract model, the parameters of the switching times and the numerical formula. In this work, however, we investigate the impact of S1, S2, and S3 on the accuracy of the

calculation for performance measures in server clusters within some different cluster sizes, namely, $K = 10; 50; 200$ servers.

We apply the service time is $1/\mu = 100s$ which is more than three times to the mean setup time and the mean shutdown time. This assumption is reasonable for a server to serve a job corresponding to such switching times. The average active power of the reference server is $P_{\max} = 56.1W$ [8].

We take six following scenarios into comparison:

Scenario	**S1 (distribution)**		**S2**	**S3**
	Setup time	**Shutdown time**		
U123	Uniform	Uniform	Yes	Yes
E123	Exponential	Exponential	Yes	Yes
U12	Uniform	No	Yes	No
E12	Exponential	No	Yes	No
U1	Uniform	No	No	No
E1	Exponential	No	No	No

Table 1: Configuration of scenarios in simulation

Simulation experiments

We built a software to simulate the abstract model. Simulation runs were performed with the confidence level of 99% and the accuracy (i.e. the ratio of the half-width of the confidence interval and the mean of collected data) is less than 0.0009.

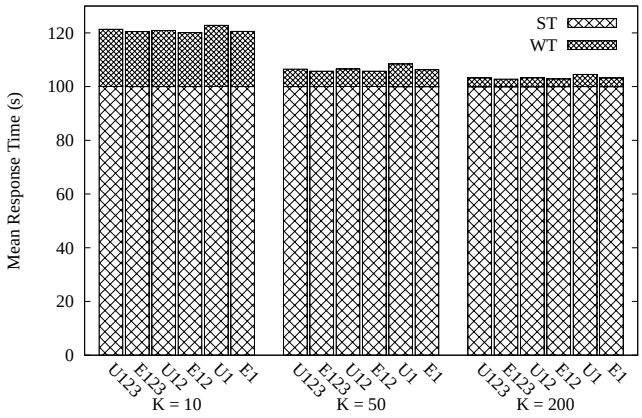


Figure 3: Mean response time vs. cluster size for $1/\mu = 100s$ and $\rho = 0.7$.

Figure 3 presents the mean response time against the cluster size for all scenarios. It is well-known that the mean response time can be decomposed into the mean waiting time (WT) and the mean service time (ST). Apparently, when the number of servers in the cluster increases, the jobs incur less waiting time. Generally, the impact of these aspects on the calculation for performance measures in terms of the mean response time is insignificant regardless the cluster size. Interestingly, the mean response time in the case of scenario **U1** is higher than that of times in **U12** and **U123** for all scenarios. S2 is the dominant aspect which causes this such phenomenon. If we do not allow turning off a server in setup process immediately (scenario **U123, U12**), there is a chance that after finishing its setup period, the server can

serve a waiting job. This job arrives during the times that server is in setup process. In other words, this job does not incur much waiting time. When we omit S2 and S3 (scenario **U1**), obviously, the arrival jobs have to wait for the full setup time which eventually increases the waiting time. Additionally, these practical aspects do not impact on the accuracy of the calculation for performance measures of server clusters under varying of cluster size.

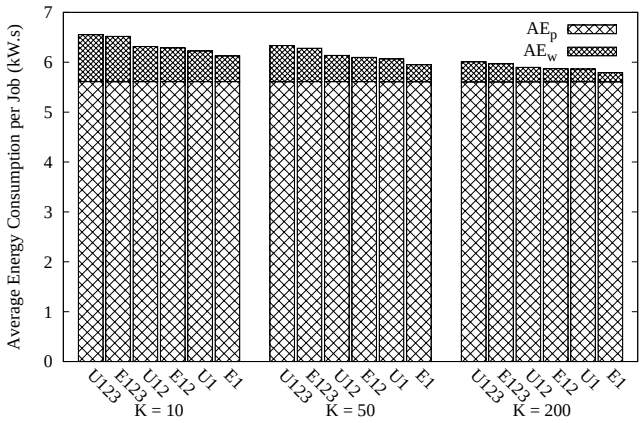


Figure 4: Average energy consumption per job vs. cluster size for $1/\mu = 100s$ and $\rho = 0.7$.

Figure 4 shows the considerable impact of these aspects on the accuracy of the calculation for average energy consumption. We define AE_p is the amount of energy that server consumed to serve job directly. AE_w is the amount of energy that server consumed in switching periods. Obviously, AE_w is wasted because it is not used to serve job at all. The impact of S2 and S3 are outperform their counterpart S1 in all scenarios in terms of the average energy consumption. In the case of small cluster size ($K = 10$), servers have to switch in and out power saving mode frequently which consumes much energy. The deviations between scenario **U123** and scenario **U12** in the case of small cluster size is 0.66 (25%), medium cluster size is 0.05 (27.6%), and large cluster size is 0.03 (27.9%). Visibly, the mismatch between cluster size and workload might causes much wasted energy consumption to the system. Another observation is the average energy consumption in models **U** is larger than that of energy consumption in models **E**. In other words, the server has longer time to consume power if the switching times is distributed uniformly.

Conclusion

In this paper, we demonstrate that:

- Although the impact of the distribution of switching times (S1) is insignificant, we should apply the uniform distribution to the switching times to improve the accuracy of the calculation for performance measures and energy consumption metrics of server clusters.
- The assumption on turning off the servers in setup process (S2) should be avoided.
- The shutdown time (S3) should be taken into account in modeling the dynamics of servers.
- These practical aspects do not impact on the accuracy of the calculation for performance measures and energy consumption metrics in server clusters under varying of cluster size.

The results in this paper might be useful to the operators who desire to evaluate the performance and to calculate the energy consumption of their server clusters. In the future, we would consider the operation policies, i.e. turn on/off a block of servers, then we investigate its impacts within practical aspects to the accuracy of the calculation for performance measures in the server clusters.

References

[1] Artalejo, J.R., Economou, A., Lopez-Herrero, M.J.: Analysis of a multiserver queue with setup times. Queueing Syst. Theory Appl. 51(1-2), 53–76 (Oct 2005), http://dx.doi.org/10.1007/s11134-005-1740-6

[2] Do, T.V., Rotter, C.: Comparison of scheduling schemes for on-demand IaaS requests. Journal of Systems and Software 85(6), 1400–1408 (2012),

[3] Gandhi, A., Harchol-Balter, M.: How data center size impacts the effectiveness of dynamic power management. In: Communication, Control, and Computing (Allerton), 2011 49th Annual Allerton Conference on. pp. 1164–1169 (Sept 2011)

[4] Tian, N., Zhang, Z.G.: Vacation Queueing Models: Theory and Applications. International Series in Operations Research & Management Science, Springer (2006)

[5] Mitrani, I.: Managing performance and power consumption in a server farm. Annals of Operations Research 202(1), 121–134 (2013), http://dx.doi.org/10.1007/s10479-011-0932-1

[6] A. Borthakur and G. Choudhury, A multiserver Poisson queue with a general startup time under N-Policy, Calcutta Statistical Association Bulletin 49 (1999) 199-213.

[7] Do N.H., Lam TB.V. (2015) Some Practical Aspects on modeling Server Clusters. In: Le Thi H., Nguyen N., Do T. (eds) Advanced Computational Methods for Knowledge Engineering. Advances in Intelligent Systems and Computing, vol 358. Springer, Cham, https://doi.org/10.1007/978-3-319-17996-4_36

[8] SPEC, Fujitsu FUJITSU Server PRIMERGY RX1330 M3 machine, https://www.spec.org/power_ssj2008/results/res2017q2/power_ssj2008-20170315-00744.html, March 2017

Bug path reduction strategies for symbolic execution

Tibor Brunner, Péter Szécsi, Zoltán Porkoláb

Abstract: Symbolic execution is one of the most powerful tools in static analysis for finding bugs. In this technique the source code is not executed by the CPU, but interpreted statement-by-statement by preserving their semantics as much as possible. During the interpretation the possible values of variables are recorded so the analyzer can report when a variable gets to an invalid state. An additional benefit of such a provided bug report is that not only the place of the bug is returned but the control flow is also displayed which results the erroneous program state. Many times the bug path contains statements which are not related to the bug. These irrelevant statements make it harder to understand in the debugging process, how the error can occur. In this article we present three methods for shortening the bug paths by removing unnecessary bug points and by recognizing two different bug paths as unique if those refer the same bug.

Introduction

During software development it is natural to make mistakes. Consequently, writing various test cases is required in order to validate the behavior of the program. In addition to the costs of test writing, it is possible that the developers fail to cover all possible critical cases. Furthermore, test writing and running often happens later than code development, but the costs of error correction increases proportionally to elapsed time [Boehm and Basili, 2001]. This proves that testing alone is not necessarily sufficient to ensure code quality.

The static analysis tools offer a different approach for code validation [Michael and Robert, 2009] [Bessey et al., 2010]. Moreover, they can potentially check for some characteristics of the code – which cannot be verified by testing – e.g. the adherence to conventions. Unfortunately, it is impossible to detect every bug only by using static analysis [Rice, 1953]. Static analyzer tools might not be able to discover some bugs (these are called false negatives) or report correct code snippets as incorrect (false positives). In the industry the goal of these tools is to keep the ratio of the false positive reports low while still being able to find real bugs.

The LLVM Clang Static Analyzer is a source code analysis tool which aims to find bugs in C, C++, and Objective-C programs using symbolic execution [King, 1975] [Hampapuram et al., 2005]. During symbolic execution, the program is being interpreted, on a function-by-function basis, without any knowledge about the runtime environment. It builds up and traverses an inner model of the execution paths, called `ExplodedGraph`, for each analyzed function.

Concepts

C++ is a compiled language which means that programs have to be transformed to a binary code for running. In static analysis techniques this phase is skipped, thus our bug finding process makes observations in compile-time. Symbolic execution is one of the most powerful techniques for bug finding as it not only considers the abstract syntax tree (AST) of the program, but also maintains the possible values of variables.

The source code is interpreted statement-by-statement while their semantics are preserved as much as possible. For example when the analyzer meets an `if` statement then the analysis is divided to two branches too. Or if a loop statement is arriving, then the analyzer also interprets the body statements with some limitations. The point is that the code is not run by CPU, but by the static interpreter. This interpreter is equipped with a constraint solver which aims to maintain the possible value range of the variables and states of composite objects. When a *checker* notices that a variable or an object is in invalid state, it can report the error to the users.

The analysis starts from certain functions. These functions which are the sources of the simulated execution are called *top level* functions. The execution path from top level functions to an error point is called a *bug path*. Along the bug path there may be some statements which are essential points concerning the bug. These statements take part in the establishment of a bug. Suppose that there is a division in the program. If the denominator is zero then an error has to be reported. The value of the denominator

expression can be assembled of several variables. Their values are set in previous statements, maybe in the body of an `if` statement. All the statements which affect the value of the denominator are so called *bug points*. The *bug length* is the number of bug points on the bug path.

Notice that this method also provides the path leading to the erroneous statement besides the place of potential error in order to help the programmer to find how the bug can arise. Along the bug path the value ranges of variables and the object states are also available, so the context is provided to the users too. These information are crucial in understanding what circumstances play role in the occurrence of a bug. The goal of the Static Analyzer is not only to report possible errors, but to cut the debugging session short.

Problems

The problem with this approach is that some bug points are also added to the bug path which are not relevant from the bug point of view. It is not helpful if the user has to inspect each step of the path and check the values of variables on certain control flows which have nothing to do with the actual bug. The appearance of such bug points is inevitable, since it might not be possible for a checker to decide whether that point is important in case of the bug.

Another problem is that there may be several bugs which can be reached from multiple places. Let us consider a function which contains a bug. There are as many potential bug paths as many paths lead to this bug from other functions. These paths are reported to the users as separate bugs which are to be fixed one by one. This is time consuming and also unnecessary, since the bug should be fixed at one place in the code base, namely in the bug container function. The situation is even worse if the bug happens in a header file. Headers are included in several translation units, so it is likely that their functions are invoked from quite a lot of locations.

Solutions

In this section we propose solutions on the above mentioned problems. For the sake of simplicity a "division by zero error" or "null pointer dereference" will illustrate the bug we intend to find and reduce its bug path length in the following examples.

Throwing unnecessary points by slicing

The first solution intends to eliminate the unnecessary statements from the bug path which are not related to the specific report. We say that a bug point is irrelevant if the variables in the final bug point are not affected by any assignment in the statements of the previous bug points. This technique is the so called backward-slicing. In our case this algorithm can be accomplished in a straightforward manner, since the list of these preceding statements is already given by Static Analyzer as the nodes of the exploded graph. These nodes contain exactly those statements and their changes which were important in some way according to the checker in the composition of the final error state. The typical example is the sequence of `if` or other conditional statements before the bug. The analyzer engine takes these conditions into account even if their *false* branch is taken and thus not related to the bug.

```
1 void f(int a, int b) {
2   int x = 0;
3   if (a < 42) { action1(); }
4   if (b > 3)  { action2(); }
5   int y = 1 / x; // Division by zero error
6 }
```

Listing 4: Code snippet, where the important statement from the view of the bug (seting the value of `x` to zero) is far from the actual bug occurrence (division by zero), however, the statements between them are not relevant.

In the example shown on Listing 4 the conditions of the branching statements will be the part of the bug path by default, assuming that the *false* branches are taken even if these are not important for the bug at all. These can be removed from the bug path in a post-processing session.

Determining unique paths by merge locations

Another category of bug shortening possibilities is finding the locations to unique the bug paths which are considered the same from a specific point. In order to combine these reports, the checkers can specify a statement which is the root of all the following errors. This common statement is called the *uniqueing location*.

```
1 void main() {
2   int x;
3   std::cin >> x;
4   int *p = nullptr;
5   // ... Not setting p
6   if (x) { *p; } // Null pointer dereference
7   else   { *p; } // Null pointer dereference
8 }
```

Listing 5: The null pointer dereference occurs at two different positions, however, logically the error is committed once, when the value of variable `p` was not set other than a null pointer.

Listing 5 demonstrates that it does not matter from which direction the null pointer dereference is reached, because the problem is that the pointer is not set correctly before the `if` statement. In the current solution of Static Analyzer, checkers can provide the specific statement, i.e. the *uniqueing location*, that resulted the error (null pointer assignment to `p` in this case), but this is always local to a translation unit. With a post-processing session we can collect all these statements for the bugs among translation units thus determining a global uniqueing location which enables us to reduce the number of separate bug paths.

Cutting a bug path prefix

By the nature of Static Analyzer the bug paths are determined from the beginning of the top level function, where the analysis started. This doesn't mean that all statements along the found bug path are needed. In practice there is a relevant point on the path which is enough to start from while debugging. For finding this point we rerun the analysis starting in the bug points from the end towards the top function. The first location from where the bug arises is enough to display.

Listing 6 shows a case where the analyzer has to split the simulation, and simulate both branches of the `if` statement since it has no information about the concrete value of `x`. This will result in two different bugs since both paths contain the error and the bug paths are different. However, the calling context does not affect the occurrence of the error. Thus, we could cut the bug path to begin from `foo` function, as the bug can be surely determined by the top level analysis of `foo`. This method not only helps the programmers to understand the code easier but also shows that the two reported bugs on Listing 6 are equivalent.

```
1  int foo(int k) {
2    int num = 42;
3    return k / (num - 42); // Division by zero error
4  }
5
6  void main() {
7    int x;
8    cin >> x;
9    if (x) foo(2);
10   else   foo(3);
11 }
```

Listing 6: Since the calling context is irrelevant in the view of the division by zero error, there is no need report it twice.

Conclusion

Clang Static Analyzer is a powerful tool for bug finding with static analysis. However, in some cases it generates unnecessarily long bug paths which are not worth to entirely review by programmers while debugging, since it contains irrelevant information from the specific bug point of view. We can introduce three methods using post-processing for shortening these paths by removing unnecessary bug points and by recognizing two different bug paths as unique if those refer the same bug.

Acknowledgement

This work is supported by the European Union, co-financed by the European Social Fund (EFOP-3.6.3-VEKOP-16-2017-00002).

References

[Bessey et al., 2010] Bessey, A., Block, K., Chelf, B., Chou, A., Fulton, B., Hallem, S., Henri-Gros, C., Kamsky, A., McPeak, S., and Engler, D. (2010). A few billion lines of code later: Using static analysis to find bugs in the real world. *Commun. ACM*, 53(2):66–75.

[Boehm and Basili, 2001] Boehm, B. and Basili, V. R. (2001). Software defect reduction top 10 list. *Computer*, 34(1):135–137.

[Hampapuram et al., 2005] Hampapuram, H., Yang, Y., and Das, M. (2005). Symbolic path simulation in path-sensitive dataflow analysis. *SIGSOFT Softw. Eng. Notes*, 31(1):52–58.

[King, 1975] King, J. C. (1975). A new approach to program testing. In *Proceedings of the international conference on Reliable software*.

[Michael and Robert, 2009] Michael, Z. and Robert, K. C. (2009). The real cost of software errors. *IEEE Security & Privacy*, 7(2):87–90.

[Rice, 1953] Rice, H. G. (1953). Classes of recursively enumerable sets and their decision problems. *Trans. Amer. Math. Soc.*, 74:358–366.

Benchmarking Graph Database Backends—What Works Well with Wikidata?

Tibor Kovács, Gábor Simon, Gergely Mezei

Abstract: Knowledge bases often utilize graphs as logical model. RDF-based knowledge bases (KB) are prime examples, as RDF (Resource Description Framework) does use graph as logical model. Graph databases are an emerging breed of NoSQL-type DBMSs (Database Management System), offering graph as the logical model. Although there are specialized databases, the so-called triple stores, for storing RDF data, graph databases can also be promising candidates for storing knowledge. In this paper, we benchmark different graph database implementations loaded with Wikidata, a real-life, large-scale knowledge base. Graph databases come in all shapes and sizes, offer different APIs and graph models. Hence we used a measurement system, that can abstract away the API differences. For the modeling aspect, we made measurements with different graph encodings previously suggested in the literature, in order to observe the impact of the encoding aspect on the overall performance.

Keywords: Graph Database, Knowledge Base, Wikidata, Benchmark

Introduction

Representing knowledge as a graph seems to be a natural choice. People even without any specialized technical or natural science knowledge often organize concepts and relations between them as nodes connected by edges. Some knowledge representation techniques also embraced this abstraction: RDF [5] represents metadata as a graph. Even the concept of knowledge graph has been floating around in the recent years, without a clear definition [6]. We use this concept aligned with [7]: an RDF graph encoding a set of knowledge.

In the DBMS world, graph as a data model is used since the dawn of database systems. As the NoSQL movement gained traction and as problem spaces with large-scale highly interconnected schemas—such as network simulation and social networks demanded, a new family of NoSQL databases emerged, replacing the key-value and document concepts with graphs. The landscape of NoSQL graph databases is in flux even today, with various graph models [13] [4], without standardized APIs, and even without a clear definition of a native graph database [12]. We use the graph database concept as a DBMS offering some graph construct (property graph, RDF graph, etc.) as logical model.

While connecting the dots above, storing knowledge represented as a graph in a database specialized to store graphs also seems a natural choice. However, one has to choose a graph database implementation first, that in turn determines the graph model and the API. Another decisive aspect is the graph encoding method. The RDF model gives a straightforward encoding for basic knowledge structures, however, there are different encoding models for reification [8], i.e. statements about statements, which is extensively used in KBs with reference management, where every statement should be backed up by external sources.

In order to help with these decisions, we selected a few graph database implementations and loaded with the same real-life, large-scale dataset, then queried with the same set of queries randomly generated from predefined query patterns. We run different measurements with different reification strategies. From the timings of the query runs, we were able to construct the performance profile of each database—encoding strategy combination.

Our research aims to determine performance characteristics of utilizing graph databases in various problem spaces. For the field of KBs, in the early phase, we worked with an algorithm-generated graph. Our initial results [10] showed counter-intuitive performance trends where more selective queries run slower than queries with more unbound values. In [9] the authors also encountered similar phenomena with a real-life dataset. Hernández et al. evaluated databases from different families, i.e. relational, graph, triplestore, and used the publicly available and collaboratively edited knowledge base Wikidata [5] as dataset.

In this phase of our research, we also used Wikidata data, but we chose the databases exclusively from the family of NoSQL graph databases.

Experimental Setting

As we wanted to compare our results with the ones described in [9], we chose to use the same January 2016 dated JSON dump as they used in their survey. For the same reason, we measured the so-called atomic-lookup queries, introduced in [9]. The basic idea behind this method is that every reified statement has five parts: a subject, a predicate, an object, a qualifier and a qualifier value. If we fix a subset of them while the others are kept variables, we get one of the 32 possible query pattern we measured in this paper.

Even though [9] introduced 4+1 representation models, we did not examine all of them. In our research, we measured three types of encoding: (i) the property graph representation which represents the qualifiers as edge properties, (ii) the standard reification, and (iii) the n-ary relation models which introduce a new node per each statement. The qualifiers are connected to this statement node, and the parts of the reified statement are connected to this node as described in [9].

Next step was to select the database implementations. The Wikidata query service is built on a customized Blazegraph [1] database engine, so we decided to select its current stable version (2.1.4). This graph database is designed to work with large RDF datasets using the standardized SPARQL query language. We chose Neo4j [11] as it is currently the most popular graph database [2]—specifically version 3.3.3. It uses the property graph model to represent the graph dataset and defines an own declarative query language, called Cypher. Our last benchmarked DBMS was the JanusGraph [3] 0.2.0. We selected this TinkerPop-based system because it is quite popular [2] and it has an active community. This database implements almost all of its functionalities through the integration of other technologies, it uses BerkeleyDB as storage, Apache Tinkerpop as graph processing engine.

We investigated other DBMSs as well, such as Grakn, OrientDB and Graph API of Azure SQL Database, but we encountered some difficulties during the modeling and loading phase in case of these systems. The main cause of the problem was that these systems hardly support having multiple different values of the same property in a node or we did not find any available description how to load them into a database.

One of our first tasks was to ensure that the DBMSs have the same runtime environment. Otherwise, the deviations in the measuring circumstances may distort the overall query timings. We achieved this by using separate virtual machines with Azure Standard E4s v3 specification for every system, the same as in [10]. Besides the operating system and the concrete DBMS, we installed the Java and .NET Core runtime environments on all VMs.

We defined a general workflow for the measurements. The 0th step was to delete all data that remained after the previous run. In the 1st phase, we transformed the decompressed JSON data to the import format of the concrete DBMS's import tool. After the transformation, we loaded the newly created dataset into the database. Finally, our tool inserted the previously random selected variable bindings into the query templates, measured the execution times and collected the mean query times.

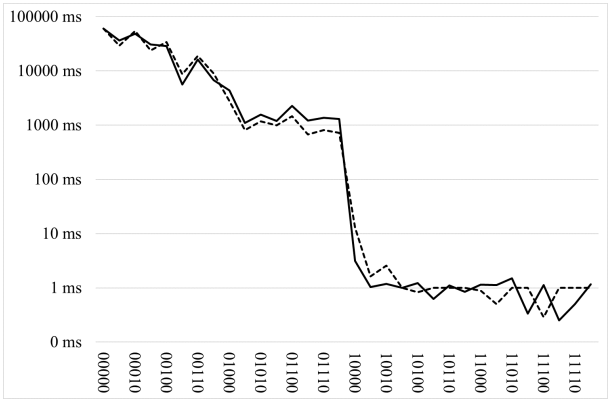
Results

We executed each 32 possible query patterns with ten different randomly selected variable bindings. To avoid first-time run transient phenomena, we ran every 320 query 2 times on every DBMS-encoding pair. We set a query time limit to 60 seconds, just like in [9] [10] to avoid it. Figure 1 shows the mean query time results in milliseconds on the logarithmical scale. Every timeout was set to 60 seconds.

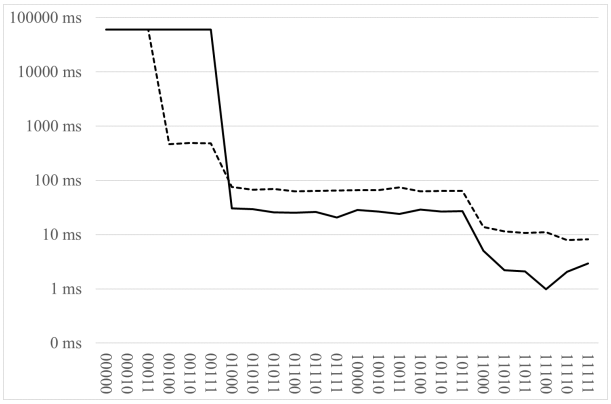
Despite its popularity, Neo4j was the slowest examined system. Every query must have been terminated before it finishes due to the time limit. Considering [9] and [10], it was expected that this system would have the slowest results, but the measured query times suggest that—despite the optimization done in the new version—this system could be several orders of magnitude times slower than its competitors on the same query, dataset and index structures.

The left diagrams of Figure 1 shows that if we used Blazegraph, there was no significant difference between the performance of the standard and the n-ary reification models. Looking at the figure, it is quite conspicuous that there is a significant gap between the times of queries with and without a variable subject. One can see that the execution times continuously decreased before and after this gap as well. This constant performance improvement can be the result of the declarative SPARQL language, whose execution can be optimized using the up-to-date DB statistics.

The other diagrams show the mean query results of the JanusGraph, and they show some similarities



Mean query times in Blazegraph using standard (solid) and n-ary (dashed) encodings.



Mean query times in JanusGraph using edge properties (solid) and n-ary (dashed) encodings.

Figure 1: Mean query times of the examined DBMS-encoding pairs.

and differences to the Blazegraph's results. The main similarity is that both system's results have steps on its diagram. The results show that in case of the JanusGraph this gap is located between the two query sets, one containing the queries without any node information binding, while every other query belongs to the other set. One can see that there is a second, smaller step at the 1 to 2 variable binding transition. The figure shows that the n-ary encoding is much faster on the first query patterns, but it is significantly slower on the later ones. Another interesting phenomenon is that the performance is almost constant between the steps, which can be explained by the imperative kind of the Gremlin query language.

Conclusions and Future Work

Regarding the mean query times, we concluded that the execution times depend heavily on both the query pattern and the system-encoding pair. The general tendency is that the less node variable a query has, the faster its execution is. The results show, even though the execution times slightly depend on the selected representation model, its impact is far less than the DBMS implementation used.

Based on the measured query times, it seems to us that the best overall performance for this kind of workload can be reached by using Blazegraph with either n-ary or standard encoding. Considering other factors than performance, our choice would be Blazegraph with n-ary representation, as this pair performed the smallest query times, while it required almost the least storage space.

We plan to involve other databases and reification techniques in our research, and replace the currently used atomic lookups to another query set that contains the most frequently used query patterns provided by the Wikidata statistics, to help the selection of the best DBMS-encoding pair for more real-life use cases.

Acknowledgments

This work was performed in the frame of FIEK_16-1-2016-0007 project, implemented with the support provided from the National Research, Development and Innovation Fund of Hungary, financed under the FIEK_16 funding scheme. Cloud computing resources were provided by a Microsoft Azure for Research award.

References

[1] Blazegraph products. https://www.blazegraph.com/product/. Accessed: 2018-03-13.

[2] Db-engines ranking of graph dbms. https://db-engines.com/en/ranking/graph+dbms. Accessed: 2018-03-13.

[3] Janusgraph. http://janusgraph.org/. Accessed: 2018-03-13.

[4] R. Angles. A comparison of current graph database models. In *Proceedings of the 2012 IEEE 28th International Conference on Data Engineering Workshops*, ICDEW '12, pages 171–177, Washington, DC, USA, 2012. IEEE Computer Society.

[5] W. W. W. Consortium et al. Rdf 1.1 concepts and abstract syntax. 2014.

[6] L. Ehrlinger and W. Wöß. Towards a definition of knowledge graphs. In *SEMANTiCS*, 2016.

[7] M. Färber, F. Bartscherer, C. Menne, and A. Rettinger. Linked data quality of dbpedia, freebase, opencyc, wikidata, and yago. *Semantic Web*, pages 1–53, 2016.

[8] D. Hernández, A. Hogan, and M. Krötzsch. Reifying RDF: what works well with Wikidata? In *Proceedings of the 11th International Workshop on Scalable Semantic Web Knowledge Base Systems (SSWS 2015)*, volume 1457 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2015.

[9] D. Hernández, A. Hogan, C. Riveros, C. Rojas, and E. Zerega. Querying wikidata: Comparing sparql, relational and graph databases. In *International Semantic Web Conference*, pages 88–103. Springer, 2016.

[10] T. Kovács. Nagyméretű szemantikus adathalmazok tárolási megoldásainak teljesítményközpontú összehasonlítása. In *BME-VIK TDK*, 2017.

[11] I. Robinson, J. Webber, and E. Eifrem. *Graph Databases*. OReilly Media, 2015.

[12] M. A. Rodriguez. A letter regarding native graph databases. https://www.datastax.com/dev/blog/a-letter-regarding-native-graph-databases, 2013.

[13] M. A. Rodriguez and P. Neubauer. Constructions from dots and lines. *CoRR*, abs/1006.2361, 2010.

Graph-based analysis of Influence Spread

Viktor Homolya

Abstract: The influence maximization is a well-known problem in network science. This problem is to target k nodes as seeds in a network G and maximize the spread of influence in this network. Lots of models have been created for this problem and to find relatively good results is easy. From the neighborhood of graph's nodes we can define local optima in the influence maximization. We aim to find connections between the structure of graph and the local optima for classifying the problem's difficulty for a given input graphs or to find properties that make easier the searching of nearly global optimal results in huge networks.

Keywords: influence spread, social network, local optima network

Introduction

The influence maximization is often used in viral marketing. In economy the problem can be interpreted as we have a limit for promoting (count of seeds) and we try to maximize the spread of information about our product. The graph represent the community. As a social network, the graph does not have multiple edges or loops. In our work the graph is directed (both way) for the different influences between people.

We can defined local optima by neighborhood from the graph. From local optima we build a network, the Local Optima Network (LON). With LON we map the codomain of original graph. We use some network science node properties (e.g. betweenness centrality, closeness centrality) and basic graph properties in special ways for the analysis.

This work made for find connections between the original graph and the created LON, which can be help in the influence maximization problem. It's not goal to optimize the problem by LON.

Influence maximization and models

The problem defined by Kempe et al. [1]. They showed the problem in the basic models (cascade models and threshold models) which they considered is NP-hard. We worked with two simple models.

Independent Cascade model (IC): every edge has a parameter p ($0 \leq p \leq 1$) representing the probability of spreading over the edge. In an iteration a freshly influenced node tries to spread to the (out)neighbors according to the edge parameter. In this model if a node v could not influence its neighbor w in first time it can't be anymore. If every influenced node can't make new influenced node(s) the model will stop.

Linear Threshold model (LT): every node and edge have a parameter t and w ($0 \leq t, w \leq 1$). The t parameter is the node's threshold value, w is the force of the influence via the edge. The sum of the incoming edges' w has to be less or equal than 1 for each node. A node v will be influenced in the next iteration if the sum of influenced incoming edges' w is bigger or equal than t value of v. The stop condition is the same as IC's.

In the work of Kempe et al. [1] the equivalence of IC models and LT models is proven. Hence, we can use only one of these models. We made tests in both models and the LT proved to be slower. So the bigger part of testing was made with IC models. From the stochastic nature of parameters a value of seed set ('influence value') came from average of influenced nodes' quantity with R times repeated evaluation.

Local Optima Network

We defined local optima from the neighborhoods of the nodes. A local optimum is a k sized subset of nodes like every solution in this problem. From these local optima created a graph, the Local Optima Network (LON). The LON approach was investigated for continuous optimization in [2], and for combinatorial optimization problems in [3]. Influence maximization belongs to the latter case, although we need to introduce some changes and new definitions. Every detected local optimum is a node in the

LON. Two nodes are connected in LON if the optimizer algorithm found them one after another. The LON is a directed, weighted graph depends on the optimizer. This optimizer has to take consideration the neighborhoods of nodes. We created a hill climbing algorithm for this task. This algorithm can traverse the edges of LON multiple times during the building phase, and this is the reason why LON is a weighted graph.

The stop condition of building is arbitrary. If we build LON for a long time we can perceive that some nodes' indegree is higher. We selected stop condition to find sufficient quantity of points with high sum of indegree. The indegree in LON characterizes the basins of a local optimum. We dropped the low weighted edges and the isolated nodes. With this filtering we created a connected graph. For analyzing the spreading in the original graph G via the LON and the filtered LON (H) we defined some measures like average of betweenness centrality of nodes from a seed set and neighbors of them in G.

In order to gain knowledge about the basins of the original problem we evaluated the closeness centrality (CC) values of nodes of H graph. The reciprocals of weights of edges were the distances in CC. With these CC values and the influence values we could classify the solvability a given problem to be difficult or easy. The high influence valued points represent nearly global optimal results. The low CC value means the point was hardly available for the hill climbing related algorithm. If we saw the most of the high influence valued points have high CC values we graded this problem to easy. Every created measure compare with influence value for find graph properties to make easier the optimization in huge networks.

Acknowledgements

I want to say gratitude to supervisor Tamás Vinkó for his helping. The project has been supported by the European Union, co-financed by the European Social Fund (EFOP-3.6.3-VEKOP-16-2017-00002).

References

[1] David Kempe, Jon Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, 2003.

[2] Tamás Vinkó and Kitti Gelle. Basin Hopping Networks of Continuous Global Optimization Problems. Central European Journal of Operations Research 25(2017) 985-1006

[3] Fabio Daolio, Marco Tomassini, Sébastien Vérel, Gabriela Ochoa Communities of Minima in Local Optima Networks of Combinatorial Spaces Physica A: Statistical Mechanics and its Applications, Elsevier, 2011, 390 (9), pp.1684 - 1694

Exploiting temporal context in 2d to 3d human pose regression

Viktor Varga, Márton Véges

Abstract: A major drawback of end-to-end image to 3d pose estimation approaches is the abscence of rich, in-the-wild image datasets with 3d human pose annotation. In this paper we show, that splitting the task and solving the subproblems of image based 2d pose estimation and 2d-to-3d coordinate regression independently is a viable approach. What is more, we present a lightweight deep learning based model to perform 2d-to-3d human body pose regression that is able to exploit temporal information and thus improve the state of the art.

Introduction

Automated understanding of human interactions in public spaces is both challenging and important. It is imperative to have the ability to reconstruct the 3-dimensional spatial model of the participants, to reliably estimate the details of such an interaction. However, usually, we only have access to monocular images or videos of the subject. In this case, human motion is perceived through a 2-dimensional projection: restoring the depth of points is an underdetermined problem. Ambiguity of the solution remains, even if we add different geometrical constraints derived from the anatomy of the human body.

In the past few years several end-to-end solutions were given to the problem of 3d pose estimation from monocular images [1]. Since image datasets with 3d pose labels are scarce and usually they were recorded in controlled environments, it becomes challenging to train end-to-end 3d pose estimators which generalize well. Another approach is to independently solve the subproblem of 2d pose estimation from images and then predict the 3d pose from the 2d coordinates. Image databases with 2d human pose annotation are abundant, including many in-the wild data, probably this is why the problem of 2d pose estimation from images has been well studied.

The ambiguity of restoring the 3d coordinates from a 2d projection may be reduced by using videos as input, or by adding temporal constraints based on the dynamics of the human body.

In our paper we present an image to 3d pose estimation pipeline, which exploits the temporal structures behind the data. To achieve this, we utilize a state-of-the-art image to 2d pose estimation software. Our contribution is the remaining part of the pipeline, namely a deep learning solution which estimates 3d pose coordinates from a series of 2d coordinates. Our method improves the state of the art by almost 15% in the former subtask.

Related Work

2d to 3d joints Lee and Chen were among the first to deal with 2d to 3d pose coordinate regression [2], interpreting the task as a binary decision problem for each limb. Several papers introduce various constraints based on the structure and dynamics of the human body: Dabral et al. [1] propose angle limits for valid limb configurations, Zhou et al. [3] move towards personalized pose estimation with their learnt limb-length ratios. Recently, Martinez et al. [4] show that lightweight, general deep learning models can outperform many complex solutions.

Exploiting temporal information Making use of the temporal context helps us to reduce noise and exploit the dynamics of the human body or the laws of physics. Zhou et al. [5] uses temporal smoothing on input 2d poses, Mehta et al. [6] penalizes velocity and acceleration.

Method

The task we aim to solve is 3d human pose estimation from 2d pose joint locations and their temporal context. Formally, we learn function $f^* = \min_f \frac{1}{TJ} \sum_{t=1}^{T} \sum_{j=1}^{J} \|f(x_{t,j}) - y_{t,j}\|_2^2$, where $x_t \in \mathbb{R}^{2J}, y_t \in \mathbb{R}^{3J}$ are 2d and 3d body poses respectively, represented by a vector of joint points. J is the number of joints predicted and T is the length of the temporal window of analysis. The input and output of our model is a series of 2d and 3d body poses.

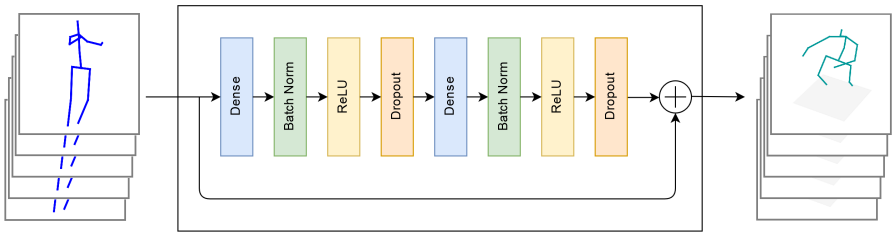


Figure 2: The architecture of our approach. The input is multiple vectors of 2d pose coordinates concatenated together. The concatenation of the corresponding 3d pose coordinates form the output of the model. The residual block at the center is repeated several times to achieve best results.

2-dimensional pose estimation We use the state-of-the-art 2d pose estimation software of Newell et al. [7] to make our pipeline capable of predicting pose in images, thus enabling the usage of the popular benchmark, the Human3.6M dataset [8] for us. We also use the 2d camera projections of the 3d pose coordinates to evaluate our 2d to 3d regression method independently.

DNN architectures Our approach exploits deep learning and its most recent results to achieve state-of-the-art prediction performance. We utilize deep neural networks with the well known ReLU nonlinearities [9]. We add batch normalization [10] and residual skip connections [11] to fight off the problem of vanishing gradients in very deep networks. Schematics of our architecture is shown in Fig. 2. This architecture was inspired by the works of Martinez et al. [4] and many others, who used similar combinations of the aforementioned layers and techniques for various purposes.

Loss function In our work, we focus on techniques that exploit temporal information and we evaluate the effect of three extra terms added to the standard L_2 loss function during the training procedure. We will refer to the first two as *Smoothness loss terms*, and define it as the mean square of the first and second order derivatives of the predictions over time. A somewhat similar term was introduced by Mehta et al. [6]. The third loss term is referred to as *Temporal limb length invariance term* which penalizes deviations of limb lengths over time in the predictions. This loss term is a novelty in its exact form, but can be derived from the work of Zhou et al. [3], who personalized predictions with fixed limb length ratios. The extra loss terms were added to the standard L_2 loss term with constant weights α, β, γ respectively.

Temporal smoothing Since our chosen 2d pose estimation software runs on single images, noise and outliers are expected to appear in its predictions. While our model may learn tasks as noise reduction or removal of outliers, we also evaluate our model on temporally smoothed input. We apply a low pass filter on the input data, to get rid of high frequency noise. Our choice is a Savitzky-Golay filter [12] fitting order 3 polynomials using a window length of 41.

Data preprocessing To prevent overfitting, we transform the 3d labels to the coordinate frame of all four cameras used in the Human3.6M dataset. Additionally, we normalize all input and output data by subtracting the mean and dividing by the standard deviation of the training data, per feature.

Results

Hyperparameter search Regarding the temporal length of the input and output, we have found that the ideal number of samples to concatenate is 8 pose vectors with a sampling frequency of 50Hz. The architecture that was found to perform best has the following parameters: a dense layer width of 4096 and consists of 4 residual blocks. Disabling residual skip connections caused serious deterioration of the results in all experiments.

3d pose regression Quantitative results are shown in Table 1. In the task of 2d to 3d pose regression using ground truth input data, our method improves the state of the art by 14.2% (6.5 mm). In case we use the 2d pose detections of the Stacked Hourglass [7] architecture, our results are beaten by Dabral et al.[1], but their model was also trained on Human3.6M image data, unlike ours, which gives their results a clear advantage.

The effect of the extra loss terms and smoothing The contribution of the loss terms were evaluated both independently and in unison. The *Smoothness loss terms* could improve 1.3 mm on average. The *Temporal limb length invariance term* improved 0.5mm on its own, and 1.5 mm together with the former terms. The following weights were found to be optimal: $\alpha = 2, \beta = 10, \gamma = 1.5$. Applying temporal smoothing on the input 2d pose data, our results were improved by a further 0.9 mm. The 95th percentile error results indicate that the amount of failed predictions were also reduced, greatly helped by input smoothing.

Table 1: Quantitative results on Human3.6M (in mm). Second column of the table shows the mean of the per-action mean joint error. Third column shows the mean of the per-joint 95th percentile errors. 2d GT: 2d projections of the 3d annotations were used as input data, SH: 2d pose input data was provided by the Stacked Hourglass network [7], 2d Tr: the 2d pose estimation model was trained or fine-tuned on the evaluation dataset unlike in our case, *: monocular approaches

	Mean error	**Mean 95th percentile error**
Martinez et al. [4] (2d GT) *	45.5	81.2
Ours (2d GT)	**39.0**	**70.5**
Martinez et al. [4] (SH) *	67.5	113.1
Zhou et al. [3] (2d Tr)	64.9	-
Dabral et al. [1] (2d Tr)	**52.1**	-
Ours (SH)	63.8	105.4
Ours (SH, smoothing)	62.7	100.9

Discussion and future work

In this paper we showed that a general deep learning approach solves the problem of 2d to 3d human pose regression effectively and can exploit temporal structures hidden in the data. Still, the surprisingly good results of Dabral et al. [1] show that an end-to-end approach may be even stronger. While the decoupling of the image-to-2d-pose estimation subproblem enables us to select our datasets from a much broader and richer palette, many precious image features, that let us infer depth, are lost.

As we have seen in the precious section, the application of a low pass filter over the input data resulted in an improvement. When we removed those samples from the Stacked Hourglass 2d pose predictions which contained clear failures (values over 100 mm in the second derivative of the time series of any joint when using a sampling rate of 20 milliseconds), the 3d pose estimations of our method improved a further 3.5 mm. We attribute this to the sensitivity of the low pass filter to extreme outliers. Instead, the application of Robust Principal Component Analysis (RPCA) [13] seems more appropriate, since this technique is less prone to highly corrupted data [14]. The investigation of this alternative is another potential future research task.

Acknowledgements

The authors are grateful to András Lőrincz, their supervisor, and Áron Fóthi, fellow researcher for their guidance and help. The project has been supported by the European Union, co-financed by the European Social Fund (EFOP-3.6.3-VEKOP-16-2017-00001).

References

[1] Dabral, R., et al. "Structure-Aware and Temporally Coherent 3D Human Pose Estimation." *arXiv preprint arXiv:1711.09250* (2017).

[2] Lee, H.-J., et al. "Determination of 3D human body postures from a single view." *Computer Vision, Graphics, and Image Processing*, 30.2 p. 148-168 (1985).

[3] Zhou, X, et al. "Weakly-supervised Transfer for 3D Human Pose Estimation in the Wild." *arXiv preprint arXiv:1704.02447* (2017).

[4] Martinez, J., et al. "A simple yet effective baseline for 3d human pose estimation." *IEEE International Conference on Computer Vision* Vol. 206, p. 2640-2649 (2017).

[5] Zhou, X., et al. "Sparseness meets deepness: 3D human pose estimation from monocular video." *IEEE conference on Computer Vision and Pattern Recognition.*, p. 4966-4975 (2016).

[6] Mehta, D., et al. "Vnect: Real-time 3d human pose estimation with a single rgb camera." *ACM Transactions on Graphics (TOG)*, 36.4: 44 (2017)

[7] Newell, A., et al. "Stacked hourglass networks for human pose estimation." *European Conference on Computer Vision*, p. 483-499 (2016).

[8] Ionescu, C., et al. "Human3. 6m: Large scale datasets and predictive methods for 3d human sensing in natural environments." *IEEE TPAMI*, p. 1325-1339 (2014).

[9] Nair, V., and Hinton, G. E.. "Rectified linear units improve restricted boltzmann machines." *Proceedings of the 27th International Conference on Machine Learning*, p. 807-814 (2010).

[10] Ioffe, S., et al. "Batch normalization: Accelerating deep network training by reducing internal covariate shift." *International Conference on Machine Learning*, p. 448-456 (2015).

[11] He, K., et al. "Deep residual learning for image recognition." *IEEE conference on Computer Vision and Pattern Recognition*, p. 770-778 (2016).

[12] Savitzky, A., and Golay. M. JE. "Smoothing and differentiation of data by simplified least squares procedures." *Analytical chemistry 36.8*, p. 1627-1639 (1964).

[13] Wright, J., et al. "Robust principal component analysis: Exact recovery of corrupted low-rank matrices via convex optimization." *Conference on Neural Information Processing Systems*, p. 2080-2088 (2009).

[14] Milacski, Z. A., et al. "Robust detection of anomalies via sparse methods." *International Conference on Neural Information Processing. Springer, Cham*, p. 419-426 (2015).

Regression Models to Predict the Resource Usage of MapReduce Application

Yangyuan Li, Tien Van DO

Abstract: MapReduce applications are used to process big data. Therefore, the prediction of the resource usage of MapReduce applications is crucially needed. In this paper, we construct multiple linear regression models to predict the resource usage parameters of MapReduce applications.

Keywords: MapReduce application; resource usage parameters; multiple linear regression

Introduction

Map/Reduce[1] is the computational programming model for processing big data. The execution of MapReduce applications may need different resource requirements. Therefore, it is important to understand the behaviour of the resource usage of MapReduce applications. L. Bautista Villalpando et al. [2] modelled the relationship between performance measurements of big data application and the quality concepts of software engineering. Issa, J A et al. [3] proposed an estimation model based on Amdahl's law regression [4] methods to estimate performance and total processing time versus different input sizes for a given processor architecture. He intended to explore the relationship between processing time and input size of data. Glushkova. et al. [5] built a new performance model for Hadoop 2.x, which use the queuing network model to capture the execution flow of a MapReduce job and take architectural changes into account. These models only concerned the performance analysis with the given resource and did not mention the factors of the allocated resource which decrease the performance of Hadoop platform. A resource reuse optimisation mechanism for MapReduce short jobs was developed by Shi et al. [6], which effectively shortened the execution time of these jobs and significantly improved the resource utilisation of a cluster. Nghiem. et al. [7] proposed a novel algorithm for optimal resource provisioning to get the exact amount of task resources, which represented the best trade-off point between performance and energy efficiency for MapReduce jobs. Bakratsas. et al. [8] evaluated the performance of three algorithms when solid state drives and hard disk drives are used to store the real social network data. However, none of previous works characterizes the resource usage of MapReduce applications.
In this paper, we establish regression models to predict the resource usage of three MapReduce applications (Wordcount, Pi and Terasort). Wordcount calculates the number of occurrence of words and the matches to a regex in a text file. The Pi application estimates the value of the Pi number, and Terasort application sorts the generated data from Teragen. The resource usage parameters (the total percentage of time spent of CPU processing job, the total memory usage, the total KB read/second from hard disk and the total KB write/second to hard disks with time resolution 1 s) of three applications are measured in the following configuration:

- Bare metal servers with an Intel CoreTM i5-4670 CPU 3.40GHz 4 cores, 16GB Kingston HyperX Black DDR3 1600MHz RAM and 250GB 7200RPM hard drive.
- Hadoop version 2.7.3 and MapReduce v2 in Ubuntu server 16.04.3 LTS, kernel 4.4.0-62-generic the block size is set to 512MB.

The workload for Wordcount is a text file of 100 GB. The workload of Terasort is 60 GB data generated from Teragen. Application Pi is executed with 2000 map tasks in 10000000 times.

Correlation Matrix

The correlation scatter matrix of Terasort application is depicted in Figure 1. All the correlations between usage parameters and their corresponding lag series show the strong positive linear relevance. The correlation between the read rate and the write rate shows the moderate negative linear relevance. Meanwhile, the correlations between read rate and lagged write rate and between the write rate and the lagged read rate exhibits the similar results. The weak negative relevance is observed in the correlation between the memory usage and the read rate.

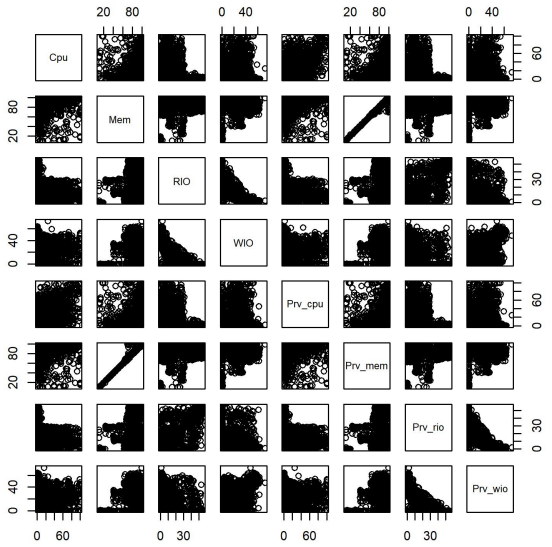


Figure 3: Scatter Matrix of resource usage parameters of Terasort

Linear Regression Models

Multiple linear regression methods are used to model resource usage parameters of MapReduce application. The entire procedure is divided into three parts in order: one is data collection; the second is filter predictors and the third is to fit model. Firstly, data collection is implemented by running Collectl, a light-weight performance monitoring tool, in parallel with the execution of applications. Typically, such kind of collected data is time series data. However, autocorrelation is usually a common characteristic of time series data. Thus, we extract the lagged usage variables which possess the largest autocorrelation coefficient for each parameter, add them to predictor dataset as well. Secondly, multicollinearity problem is taken into account and removed to subject to the important assumption of independence assumption of predictors. Furthermore, the best-subset method and ten-fold cross-validation are applied to filter predictors for regression models. Finally, the least squares methods are used to estimate the regression coefficients which is shown in Table 1.
Table 1 presents the multiple linear regression models and the corresponding residual standard error (RSE) and R^2 for three MapReduce application and regression models. Note that the Z-score (standardized coefficient), a statistical measure, is used to drop irrelevant variable from the set of predictors.

In Table 1, the estimated coefficient shows the strength of linear dependency and its sign represents the dependent direction. Except for the dependency between response and itself previous usage parameter, others dependency exposes the resource bottleneck of the corresponding application in Hadoop MapReduce environment. Meanwhile, according to the estimated coefficient, the optimized suggestion could be given for improving associated usage parameters. For example, read rate, denoted by RIO, in the model $\widehat{CPU} \sim 22.37 + 0.4 \times RIO + 0.6 \times Prv\_cpu$ of Wordcount application has an estimated coefficient 0.4. It means that the average CPU usage might increase 4% when the average read rate increase 10MB/S as well as the corresponding previous CPU usage keeps the fixed value.

Conclusion and Future

We have applied linear regression models to predict the resource usage parameters of MapReduce applications. The regression models could be beneficial to cloud operators in the assignment of MapReduce tasks in the cloud computing platform. In future, we will study the relationship between the stability of model and sampling time, the influence coming from the block size and scale of workload as well.

Name of application	Regression Model	RSE	R^2
Wordcount	$\widehat{CPU} \sim 22.37 + 0.4 \times RIO + 0.6 \times Prv\_cpu$	13.41	39.5%
	$\widehat{MEM} \sim 0.37 + 0.996 \times Prv\_mem$	0.36	99.9%
	$\widehat{RIO} \sim 11 - 6.12 \times WIO + 0.6 \times Prv\_rio$	3.70	38.9%
	$\widehat{WIO} \sim 0.04 + 0.41 \times Prv\_wio$	0.12	17.1%
Terasort	$\widehat{CPU} \sim 2.18 + 0.69 \times Prv\_cpu$	9.41	47.8%
	$\widehat{MEM} \sim 0.77 + 0.99 \times Prv\_mem$	0.89	99%
	$\widehat{RIO} \sim 1.25 - 0.24 \times WIO + 0.96 \times Prv\_rio + 0.27 \times Prv\_wio - 0.01 \times WIO : Prv\_rio$	3.19	90.4%
	$\widehat{WIO} \sim 3.01 + 1.11 \times Prv\_rio - 1.16 \times RIO + 0.93 \times Prv\_wio - 0.01 \times RIO : Prv\_wio$	6.39	83.7%
Pi	$\widehat{CPU} \sim 1.25 + 0.98 \times Prv\_cpu$	7	96.8%
	$\widehat{MEM} \sim 6.85 + 0.07 \times CPU + 0.31 \times Prv\_mem$	1.14	92.8%
	$\widehat{RIO} \sim 0.004 + 0.72 \times Prv\_rio$	0.13	54.2%
	$\widehat{WIO} \sim 0.10 + 0.36 \times Prv\_wio$	0.28	13%

Table 1: List of regression models

References

[1] V. K. Vavilapalli,A. C. Murthy,C. Douglas,S. Agarwal,M. Konar,R. Evans,T. Graves,J. Lowe,H. Shah,S. Seth,B. Saha,C. Curino, O. O'Malley, S. Radia, B. Reed, and E. Baldeschwieler. Yet Another Resource Negotiator, *Apache Hadoop YARN* , in Proceedings of the 4th Annual Symposium on Cloud Computing,p.5:1–5:16, 2013.

[2] L. Bautista Villalpando, A. April, and A. Abran. Performance analysis model for big data applications in cloud computing, *Journal of Cloud Computing: Advances, Systems and Applications*, vol.3, no. 1, pp. 19-38, 2014.

[3] J. A. Issa. Performance Evaluation and Estimation Model Using Regression Method for Hadoop WordCount, IEEE Access,vol. 3, pp. 2784-2793, 2015.

[4] D. P. Rodgers. Improvements in multiprocessor system design, *ACM SIGARCH Computer Architecture News*, vol. 13, no. 3, pp. 225-231, 1985.

[5] D. Glushkova, P. Jovanovic, and A. Abelló. Mapreduce performance model for Hadoop 2.x, *Information Systems*, 2017.

[6] Y. Shi, K. Zhang, L. Cui, L. Liu, Y. Zheng, S. Zhang, and H. Yu. MapReduce short jobs optimization based on resource reuse, *Microprocessors and Microsystems*,vol. 47, no. Part A, pp. 178-187, 2016.

[7] P. P. Nghiem and S. M. Figueira. Towards efficient resource provisioning in MapReduce, *Journal of Parallel and Distributed Computing*,vol. 95, no. Supplement C, pp. 29-41, 2016.

[8] M. Bakratsas, P. Basaras, D. Katsaros, and L. Tassiulas. Hadoop MapReduce Performance on SSDs for Analyzing Social Networks, *Big Data Research*, vol. 1, pp. 1-10, 2017.

Long Short-term Memory Recurrent Neural Networks Models to Forecast the Resource Usage of MapReduce Applications

Yangyuan Li, Tien Van DO

Abstract: The forecasting of the resource usage of MapReduce applications plays an important role in the operation of cloud infrastructure. In this paper, we apply long short-term memory recurrent neural networks to predict the resource usage of three representative MapReduce applications. The Results show that the Long Short-term Memory Recurrent Neural Networks models perform higher prediction accuracy than persistence ones. Predictions of other usage parameters show similar accuracy with persistence one. The improper configuration parameters of Long Short-term Memory Recurrent Neural Networks possibly result in few of worse prediction.

Keywords: MapReduce application; resource usage parameters; LSTM-RNN model; forecasting

Introduction

MapReduce applications are developed to process big data [1,2] in public clouds and private clouds. Therefore, forecasting the resource usage of MapReduce application is crucially needed for cloud operators. Yan Ling et al. [3] predicted the total execution time for MapReduce applications with linear regression model and correction neural network model. Issa, J A et al. [4] proposed an estimation model to estimate total processing time versus different input sizes under a given processor architecture. H. Yang et al. [5] predicted the total execution time of a workload under different Hadoop configurations with support vector regression models. However, most of them only estimated the total execution time of MapReduce jobs.

In this paper, we apply multivariate long-short term memory recurrent neural networks (LSTM-RNN) [6] to forecast resource usage parameters (CPU usage(%), memory usage(%), read rate(MB/S) and write rate(MB/S)) of three MapReduce benchmark applications. LSTM-RNN is an evolutional version of recurrent neural networks that can effectively avoid gradient vanishing and exploding. The LSTM unit is composed of a memory cell state, an input gate, an output gate, and a forget gate. Moreover, the structure of LSTM-RNN is capable to learn long-term dependencies of time series data. We use LSTM-RNN to predict the resource usage of three applications (Wordmean, Grep, and Teragen). The first two applications calculate the average length of words and the matches to a regex in a text file, respectively.

The prediction with LSTMs Models

We use LSTM-RNN models with one hidden LSTM layer and one output layer. To choose a suitable configuration of LSTM-RNN, we identified the hyper parameters (epoch size, batch size, neurons number, time steps) by performing the tuning configuration prior to the training and the forecasting activity. Then we apply a one-shot method [7].

The experimental datasets of three applications are collected from the following scenario:

- Bare metal servers with an Intel CoreTM i5-4670 CPU 3.40GHz 4 cores, 16GB Kingston HyperX Black DDR3 1600MHz RAM and 250GB 7200RPM hard drive.
- Hadoop version 2.7.3 and MapReduce v2 in Ubuntu server 16.04.3 LTS, kernel 4.4.0-62-generic the block size is set to 512MB.

The root mean squared error (RMSE) [8] is used to evaluate the accuracy of prediction as it punishes large errors and results in a score that is in the same units as the forecast data. We establish a baseline performance for each usage parameter by developing a persistence model that provides a lower acceptable bound of performance on the test set. RMSE with the use of the persistence forecast (naive forecast) and LSTM-RNN forecast on the dataset is presented in Table 1.

The results show that the intensive resource usage parameters are able to obtain better predictive performance after using LSTM-GNN models.

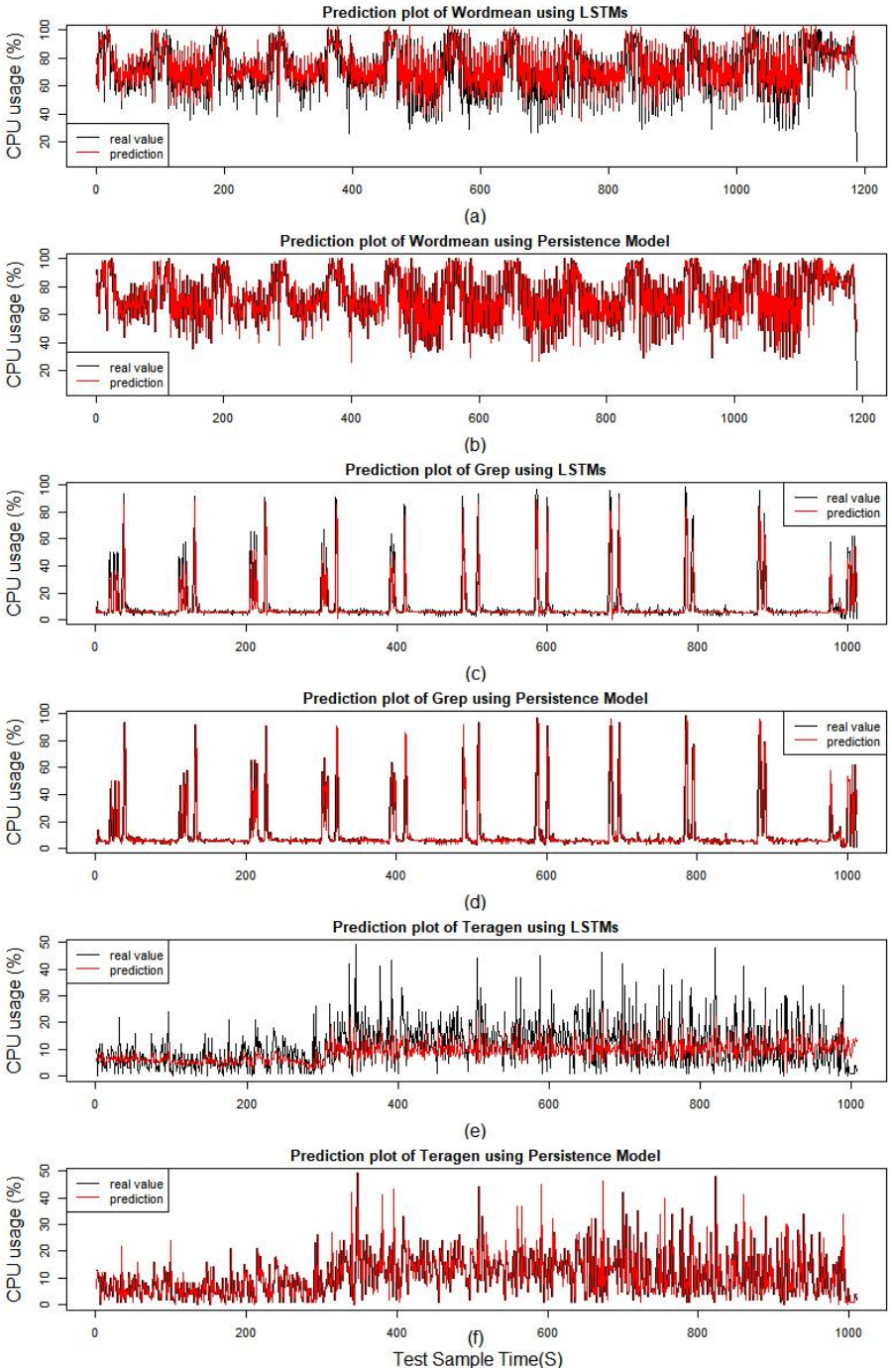


Figure 1: Forecasting comparison time series plot

Modeling Method /Application name	RMSE of CPU	RMSE of Memory	RMSE of Read rate	RMSE of Write rate
LSTM-RNN /Wordmean	14.133	0.371	2.608	0.137
Persistence model / Wordmean	25.080	0.386	3.120	0.207
Improvement rate after using LSTM-RNN	43.65%	3.89%	16.41%	33.82%
LSTM-RNN / Grep	12.492	0.393	2.889	0.120
Persistence model / Grep	13.581	0.406	2.700	0.166
Improvement rate after using LSTM-RNN	8.02%	3.20%	-7.00%	27.71%
LSTM-RNN / Teragen	8.198	0.103	0.055	5.986
Persistence model / Teragen	10.410	0.090	0.051	8.013
Improvement rate after using LSTM-RNN	21.25%	-14.44%	-7.84%	25.30%

Table 1: Prediction accuracy comparison

We draw the forecasting time series plot of CPU usage parameters between real value and prediction in Figure 1 to exhibit the forecasting performance of LSTM-RNN models. In Figure 1, the sub-figure [(a), (b)], [(c), (d)], and [(e), (f)] are used to show CPU usage time series comparison plot for Wordmean, Grep, and Teragen application respectively using LSTM-RNN models and persistence models. In Figure 1, the CPU usage forecasts of three applications with LSTM-RNN models show higher accuracy than predictions with persistence models. The persistence models only shift to right side 1 time-step.

Conclusions

We have applied LSTM-RNN model to forecast the usage parameters of MapReduce applications. The LSTM-RNN models show higher forecast accuracy than persistence models for the CPU usage prediction. The forecast accuracy for the rest of usage parameters show similar results with persistence models. Few of usage prediction get worse result possibly due to the improper configuration parameters of Long Short-term Memory Recurrent Neural Networks.

References

[1] V. K. Vavilapalli,A. C. Murthy,C. Douglas,S. Agarwal,M. Konar,R. Evans,T. Graves,J. Lowe,H. Shah,S. Seth,B. Saha,C. Curino, O. O'Malley, S. Radia, B. Reed, and E. Baldeschwieler. Yet Another Resource Negotiator, *Apache Hadoop YARN* , in Proceedings of the 4th Annual Symposium on Cloud Computing,p.5:1–5:16, 2013.

[2] A. S. Foundation, Apache hadoop. http://apache.hadoop.org. p. Last Published: 12/18/2017, 2017.

[3] Y. Ling, F. Liu, Y. Qiu, and J. Zhao. Prediction of total execution time for MapReduce applications, in 6th International Conference on Information Science and Technology, ICIST 2016, 2016, pp. 341-345.

[4] J. A. Issa. Performance Evaluation and Estimation Model Using Regression Method for Hadoop WordCount, IEEE Access, vol. 3, pp. 2784-2793, 2015.

[5] H. Yang, Z. Luan, W. Li, and D. Qian. MapReduce workload modeling with statistical approach,*Journal of Grid Computing*, vol. 10, no. 2, pp. 279-310, 2012.

[6] S. Hochreiter and J. Urgen Schmidhuber. LONG SHORT-TERM MEMORY, *Neural Computation*, vol. 9, no. 8, pp. 1735-1780, 1997.

[7] S. Ta'asan, G. Kuruvila, and M. D. Salas. Aerodynamic design and optimization in one shot, in 30th Aerospace Sciences Meeting and Exhibit, 1992.

[8] T. Chai and R. R. Draxler. Root mean square error (RMSE) or mean absolute error (MAE)? - Arguments against avoiding RMSE in the literature, *Geoscientific Model Development*, vol. 7, no. 3, pp. 1247-1250, 2014.

Full-stack FHIR-based MBaaS with Server- and Client-side Caching Capable WebDAO

Zoltán Richárd Jánki, Vilmos Bilicki

Abstract: In healthcare systems, it is essential to have applications that are robust responsive and have a good performance. It is also advisable to store data in some standardized way so it can be integrated with other systems. However, in the 21st century an application may be doubtful of use if the user-experience is at a low level. Several studies inform us about how tolerant the users are when they visit a website or wait to retrieve some data. Based on these studies, we will construct a system that is capable of working offline and can also unburden the server-side. This will be achieved by establishing a so-called Web Data Access Object (WebDAO), which has a maintainable offline capability and also performs better in most given circumstances. Our measurements were evaluated in the context of how users tolerate a delay and slow responses.

Keywords: full-stack, caching, FHIR, WebDAO

Introduction

In the healthcare sector, more and more information about patients has to be processed quickly and efficiently. As a consequence, the healthcare records must be handled in a digitized format. These electronic healthcare records (EHR) must be available, discoverable and understandable. For these requirements a standardized and structured storage is essential. The Health Level Seven (HL7) organization [6] solved these challenges by creating the so-called Fast Healthcare Interoperability Resources (FHIR) standards. With its well-defined resources, this standard is suitable for modelling any areas of healthcare field.

Using the FHIR in Web applications, it is not obvious which implementation would be the best in terms of availability and performance. Full-stack development is one of the most popular approaches today. In a full-stack environment the code base can be shared between the client and the server because they are written in the same programming language. There are open-source implementations of FHIR, but none of them supports client-side object modelling in JavaScript. Hence, we had to create our own Mobile Backend as a Service (MBaaS) following the Representational State Transfer (REST) paradigm [2]. This paradigm asserts that neither the server, nor the client knows anything about the other's state. This stateless system allows both participants to handle the messages received from the other one without seeing any messages that arrived earlier.

To improve the performance, we decided to pursue the idea of polyglot persistence where we use different database systems, but each of them is used for what they are best at. In order to achieve the highest availability we have to exploit the offline capabilities of the client-side and the server-side and create offline-first applications. The framework called Hibernate applies the Data Access Object (DAO) Pattern to separate low-level operations from high-level business services. It uses its own caching model where the queries are forwarded to the different levels of the cache first instead of communicating with the backend. The Hibernate Query Language (HQL) is first forwarded to the Second-Level (L2) Cache and it searches for the queried objects in it. If the required objects are not in the L2 Cache, the request is forwarded to the database. By doing this, the waiting time for an application can be significantly reduced [3][4].

There are several studies available concerning the correlation betweeen users' patience and speed of the Web. As Daniel An from Google said in his article, mobile pages can display content to users within 3 seconds on average and the average time to first byte is under 1.3 seconds [5].

Xiaming Chen and et al. showed that if the time between the emission time of the request and the arrival of the response is more than 4 seconds, the interruption ratio rises rapidly from 18%. If a waiting time of 10 seconds is reached, the ratio of interruption is already at 40% [6]. We use these thresholds to provide a solution that is quite effective.

Below, we are going to review the different offline-first solutions in a full-stack environment and then we would like to present our solution. We will describe the novel features of our system, present the results of our measurements, then draw some conclusions about our prototype systems.

State of the art

FHIR and full-stack development

Over time, FHIR has become widely acknowledged. Several organizations use this standard and some of them are so big that they are present in many countries. For instance, the InterSystems HealthShare [7] is a healthcare informatics platform for hospitals, integrated delivery networks and regional and national health information exchange. It is now present in more than 20 countries. Another well-known project is the so-called Substitutable Medical Applications and Reusable Technologies (SMART) [9]. Since it uses the FHIR standard, the platform was renamed to SMART on FHIR. These systems are widely spread, but none of them uses a full-stack environment.

However, today JavaScript is the only programming language that is supported by every major browser. As we review the history of JavaScript, we see that over time it underwent numerous changes and we can say that JavaScript is a language that is suitable for any general-purpose computing task. [9].

Offline Capabilities

When we talk about Web applications, everybody thinks of online applications but what if there is simply no Internet connection or it is Lie-Fi. Such situations can also occur in mobile applications that require a network connection. In all of the applications, the developers have to take into account network issues. In healthcare systems, availability and performance are two key factors. Can we have an observation that is interrupted because of a bad network connection or is it necessary to query all the records of a patient again after a refresh? The answer is of course no. This is why the offline capabilities of Web applications are very important for us. The basic methodology of caching in Web applications follows the idea of Hypertext Transfer Protocol (HTTP) Caching, but using a framework, this layer is totally hidden. Nowadays, on the client-side the recommended solutions are the use of Service Workers and IndexedDB. These tools guarantee convenient solutions for using the cache storage of a browser.

HTTP Cache

Every major browser carries an implementation of a HTTP Cache. This methodology is based on a proxy-server - also called a Web cache - that meets HTTP requests on the behalf of an origin Web server. The cache storage has its own disk storage that includes the copies of recently requested objects. The HTTP headers are extended with a Cache-Control field and an ETag field [10][11]. These fields are responsible for defining the caching policy and validating the response of the server. Service Workers and IndexedDB use the notion of a HTTP Cache and are involved in developing progressive Web applications.

Service Workers and IndexedDB

A Service Worker is a Web Worker object that is a JavaScript file running in a worker thread. Hence, it is separated from the browser's main thread and it can be executed asynchronously. Before Service Workers, the recommended storing mechanism was the Application Cache – also known as AppCache. AppCache provides a high-level API and every browser has support for it. Since it was not sufficiently flexible, AppCache was replaced by Service Worker, with its low-level Cache API. It is more adaptable because it hands over the "moving parts" to the developers and the configurations are left to the developers [12].

IndexedDB also provides a low-level API for caching data in a NoSQL storage system. It creates a sterling database in the browser and supports transactions as well. The objects can be stored and searched by key-value pairs. It works along similar lines to the local storage of the applications, but IndexedDB works asynchronously. It runs in the background and the background synchronization is also supported [13].

Our solution

A telemedicine application must be able to handle thousands of patients and hundreds of doctors with their data simultaneously. It should be responsive and have a high performance if the architecture is well defined and robust. Nowadays, full-stack development can achieve these requirements. On the client-side we use Angular 2+ frameworks and the backend consists of a LoopBack Server and several databases that realize the polyglot persistence. The Hadoop HDFS is used for storing huge files like videos and images, and Apache Cassandra is responsible for the patients' big data.

On the client-side it seems clear that Service Workers and IndexedDB have to be used. The offline capabilities of LoopBack Server are quite limited and there are no best practices for caching in LoopBack Framework, so we made some extensions to the basic architecture.

Firstly, we created a WebDAO that follows the classic DAO analogue. In our WebDAO layer we defined the models of our resources and the basic methods of the models. With the help of this layer, we were able to write queries on both the client-side and the server-side. Hence, the Cache Storage is not just a plain cache. The programmers can maintain the offline activities and also the response time may be better in the given circumstances. The main advantage of WebDAO is that we do not have to create our own data structure for storage, but we can use the HTTP Cache on the client-side.

IndexedDB can be actuated by another JavaScript database called Apache PouchDB. PouchDB is an open-source database that can synchronize with CouchDB and compatible servers. PouchDB also has a browser version that can be readily integrated into our Angular clients. It provides the toolset for creating an IndexedDB instance and also for modifying and synchronizing its content. With this technology we do not have to manipulate IndexedDB directly. Hence, our idea is to store the static objects - like images and HTML parts - in the Cache Storage by using Service Workers, and store the dynamic parts - like records from the Cassandra database - in an IndexedDB instance. The contents of PouchDB will be synchronized with the CouchDB instance that is also synchronized with LoopBack. These frameworks can improve the availability and the performance as well because if our application goes offline, the data is still available from the cache and using an offline-first strategy, we will not turn to the server for each requests, except if the data is unavailable in the cache. With the help of PouchDB, we can also use filters in our cached dataset, so a new request with filters will not take another request to the server. Another good thing about PouchDB is that it can communicate with an IndexedDB instance in the development mode, while native Service Workers and IndexedDB require a production mode for caching the responses.

Measurements and evaluation

In our system we created a so-called Web Data Access Object (WebDAO) which defines the model of FHIR resources, and these models have their own standardized REST endpoints. By following the REST paradigm, our WebDAO became general and in order to unburden the backend, we entrusted the caching to the browser. Since the REST endpoints are more general, the response is also broader, hence more specific queries can be handled on the client-side by retrieving objects from the Cache Storage. The exact time values can be calculated by the browser, since it can differentiate a query result received from cache or from a server.

Since the whole infrastructure takes place in our local network, the server response time can be much higher in production where we can have different network facilities and overloads. Our dataset contains 1 patient and 1,000,000 different observations belonging to this patient, but the Apache Cassandra retrieves at most 5,000 records in a query because of its driver setup. The first query that we executed was very simple. We queried all of the observations that belonged to our single user. The time that elapsed between the emission of the request and the arrival of the response can be seen in Table 1. It is observed that pulling 5,000 records through the network takes more than 1 second and we can say that it is worthwhile storing the objects in the cache and using an offline-first strategy. If we use more than one filter, the Cassandra database is still fast and the set of objects is much smaller, so a response can be obtained in 300 ms. PouchDB with IndexedDB seems to be weaker here because PouchDB is optimized for synching first. The queries can be speeded up by using indices. If we do not use any indices, the filtering costs more than 5 seconds. This is too much, so using indices is recommended. With indices the request takes about 500 ms. We also tried out filtering with our own indices by creating references from the proper keys in an array. The result was very similar and the difference was about plus or minus 10

ms.

Table 1: COMPARISON OF RESPONSES RECEIVED FROM SERVER AND CACHE

Request	*Query*	*Source of response*	*Number of retrieved records*	*Average time to get response*
List Patient1's observations	user: Patient1, date1: null, date2: null	LoopBack and Cassandra	5,000	1159 ms
		IndexedDB using PouchDB (no-filtering)		521.75 ms
List Patient1's observations that occured between date1 and date2	user: Patient1, date1: 2005-09-07T01:00:00.000Z, date2: 2006-09-07T01:00:00.000Z	LoopBack and Cassandra	8	291 ms
		IndexedDB using PouchDB filtering (without indices)		5446 ms
		IndexedDB using PouchDB filtering (with indices)		519.25 ms

Comparing these results with those that were presented in the above-mentioned studies about the general patience of users, we see that, our system still works within the thresholds and the applications integrated into the system can work offline as well.

Conclusions

Here, we presented a system that uses an offline-first strategy in order to achieve a higher performance and enhance availability. We created a WebDAO that generalizes the client-server communication and makes the tool available to the programmer to maintain the Cache Storage. With polyglot persistence our backend is now more powerful and the whole system works with a similar efficiency in the online state as in the offline state. In the future, we intend to improve the performance of the Cache Storage by using other indexing schemes like indices on patient data and other observation properties.

Acknowledgement

This study was supported by the EU-funded Hungarian grant EFOP-3.6.1-16-2016-00008.

References

[1] *FHIR Overview*, Available: https://www.hl7.org/fhir/overview.html, Accessed: 21 March 2018

[2] B. Mulloy, *Web API Design: Crafting Interfaces that Developers Love*, Available: https://pages.apigee.com/rs/apigee/images/api-design-ebook-2012-03.pdf, Accessed: 28 March 2018

[3] J. P. Ottinger, D. Minter and J. Linwood *Beginning Hibernate*, 3rd Edition, 2014, Apress, United States

[4] J. P. Ottinger, S. Guruzu and G. Mak *Hibernate Recipes: A Problem-Solution Approach*, 2nd Edition, 2015, Apress, United States

[5] D. An, *Find out how you stack up to new industry benchmarks for mobile page speed*, Available: https://www.thinkwithgoogle.com/marketing-resources/data-measurement/mobile-page-speed-new-industry-benchmarks, Accessed: 28 March 2018

[6] X. Chen and et al., *Passive profiling of mobile engaging behaviours via user-end application performance assessment*, Pervasive and Mobile Computing, vol. 29, 2016, pp. 95–112

[7] *InterSystems Health Informatics Platform*, Available: www.intersystems.com/products/healthshare/intersystems-health-informatics-platform, Accessed: 21 March 2018

[8] J. C. Mandel and et al. *SMART on FHIR: a standards-based, interoperable apps platform for electronic health records*, Journal of the American Medical Informatics Association, vol. 23, no. 5, 1 September 2016, pp. 899–908

[9] A. Bretz and C. J. Ihrig, *Full Stack JavaScript Development with MEAN*, 2014, SitePoint Pty. Ltd., 48 Cambridge Street Collingwood VIC Australia 3066

[10] J. F. Kurose and K. W. Ross, *Computer Networking: A Top-Down Approach*, 7th Edition, 2017, Pearson Education Limited, Edinburgh Gate, Harlow, Essex CM20 2JE, England

[11] I. Grigorik, *HTTP Caching*, Available: https://developers.google.com/web/fundamentals/performance/optimizing-content-efficiency/http-caching, Accessed: 22 March 2018

[12] *HTML Living Standard*, Available: https://html.spec.whatwg.org/multipage/offline.html, Accessed: 26 March 2018

[13] P. Walton, *Best Practices for Using IndexedDB*, Available: https://developers.google.com/web/fundamentals/instant-and-offline/web-storage/indexeddb-best-practices, Accessed: 26 March 2018

A FHIR-based healthcare system backend with deep cloud side security

Zoltán Szabó, Vilmos Bilicki

Abstract: The need for an easy-to-learn healthcare development framework is becoming evermore urgent, but the challenges created by the very specific nature of this domain, combined with increasingly complex technological requirements is no easy task. In this paper, we introduce our proposition for one such development stack, ready for handling large amounts of medical data and security issues. Then we introduce benchmark-based evaluations on a heavily loaded database instance, and compare them with a similar benchmark from another team, who used a similar approach with the FHIR standard but with different technologies.

Keywords: Apache Cassandra, FHIR, full-stack, security, healthcare

Introduction

The area of e-health is rapidly evolving. The focus of our developer team is not just to create a full-stack development framework that provides a simple, easy-to-learn solution for every developer who wants to enter the area of IT Health Systems, but also to make the resulting applications cloud-compatible, failure tolerant and 7/24 available. In order to establish a system like this, we not only had to face the usual difficulties of IT Health System development, such as the need for strict, yet dynamic security handling for real-life situations, but also the lack of options for some specific requirements of our system.

Our framework was written in a combination of JavaScript and TypeScript, with the server-side using the popular LoopBack framework [1] and the clients written in the Angular [2] and its extension the Ionic [3] framework. To store the healthcare data, we implemented the concept of polyglot persistence, using multiple cloud-based databases based on the areas and use cases they were designed for, in order to optimize speed, security and space requirements, while also taking the concept of cloud computing in account. Accordingly, the majority of documents are stored on an Apache Cassandra [4] cluster, while video and voice recordings from the patients and doctors are maintained in the Apache Hadoop File System [5]. We also needed a unified, officially accepted data structure, hence we chose the Fast Healthcare Interoperability Resources (FHIR) standard from the Health Level 7 (HL7) organization [6]. However, the use of the FHIR standard raised many questions for our development team, the most prominent being the issue of security. The FHIR standard provides only guidelines for implementing authorization mechanisms, such as the use of security labels as part of the meta information in the resource document and ideas to implement either the Role-Based Access Control, where the role attribute of the current user determines the enabled operations or the Attribute-Based Access Control, where the access depends on the value of an attribute in the resource [8].

For our server side, we also wanted to use the OpenAPI specification to provide our current and future developers with a simple, standardized way of creating the API endpoints. LoopBack is a readily extendable, OpenAPI-compatible Node.js framework with an emphasis on the easy creation of dynamic end-to-end REST APIs and a very supportive developer community. We used some of these community-created solutions to integrate the FHIR HAPI server [7], a popular, Java-based implementation of the standard into our stack as a database, namely the loopback-rest-connector and the loopback-sdk-builder to generate client modules for our Angular Framework codebase.

This kind of integration however leads to a rather slow backend, where the potential developers lacked full control over a key part of the data storage in the form of the HAPI server. We found that other developers either integrated the FHIR HAPI server into their developments in a way similar to our early prototype and the Smart on FHIR project [9], or went for an incomplete implementation in their own development environments like the hospitalrun.io project [10].

After this research cycle we planned a new approach, which would give us complete control over the data and the domain, is cloud-compatible, has a security module providing authorization of clinical data in real-life use cases, and even under a heavy load is at least as reliable as those in the studies we assessed during our design phase.

Our solution

Cassandra

The first step was to transfer of our data model into the Apache Cassandra database. During development, like in the hospitalrun project, we found it unnecessary to use and implement the entire set of the FHIR resources. The five components we needed to refactor for our already existing projects and solutions based on the FHIR standard were Patient, Practitioner, Observation, DetectedIssue and Location.

We also decided to use the unique indexing structure of Cassandra to optimize the partitioning and the order based on our most common queries. In our Observation model for example we chose the id for the primary key and defined the clustering order by the *effectiveDateTime* field, which records the exact timestamp when the given observation or measurement was taken, then the *code*, representing the LOINC [11] code for identifying the measurement type. These three are also the most common query parameters used in almost every query we wrote for the HAPI server when retrieving the data.

LoopBack Server

After we had succeeded in finding a good data modeling strategy, next we had to construct a LoopBack server to handle the client requests to the Cassandra database. For this, our team used the popular OpenAPI 2.0 [12] specification to describe every REST endpoint needed by our current developments. The server code was generated from this specification with the Swagger generator module, with the business logic of the endpoints later included and tested.

After completing this server, we generated the SDK modules with the loopback-sdk-builder for our TypeScript-based Angular clients; and these were successfully built into our client projects by our junior developers, replacing the vanilla HTTP calls made directly towards the FHIR HAPI server with socket-based calls towards the LoopBack server.

Security Module

The remaining issue was of course that of the security. For this domain, we designed special APM tables that described the connection among the potential applications, users and data types. A typical row in this APM database consists of the *key* field, which describes the owner of the data, the type of the data, and the application requesting the usage of the data, the *read*, *write* and *authorize* containing the identifiers of the users who are authorized to read, write or share the given resource, and finally, the *startDate* and the *endDate* timestamps signifying the valid period of the given entry. The APM database is loaded into the memory when the server is booted up.

When querying the documents, we only send the basic query parameters to the Cassandra database, such as the patient ID, the restrictions on effectiveDateTime and the code. After the LoopBack server receives a dataset, it forwards it to a filter module, which makes use of the APM records to it. This way the dataset received by the clients contains only those documents to which the authorization filter found an APM entry declaring effective authorized access.

Evaluation

After the successful implementation, next we had to benchmark our solution. When analyzing our results, we used the measurements taken from the FHIRBase project [13] for comparison, which released various benchmark results, where a PostgreSQL database held the FHIR documents. We established a test server with similar parameters to the Amazon EC2 Instance Type m3.large the FHIRBase developers used for their tests. Although our implementation had no Encounter resource, their implementation of Encounter was similar to how we used the Observation resource, in terms of the quantity of the stored information and the most common query parameters (Patient ID, Practitioner ID, effectiveDateTime, etc.)

Our results are shown in figures 1-4. While running the benchmark, the database was loaded with exactly one million Observation documents that belonged to multiple patients. The benchmark was accomplished with a lightweight Node.js application, which increased the query limit for each step,

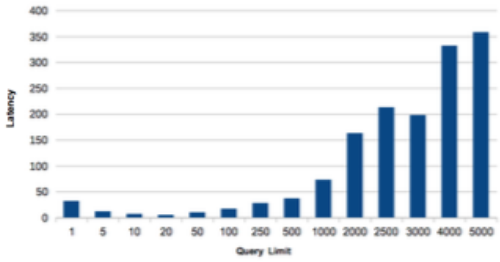


Figure 1: The query latencies when just the Patient ID is used as a filter, without any security restrictions

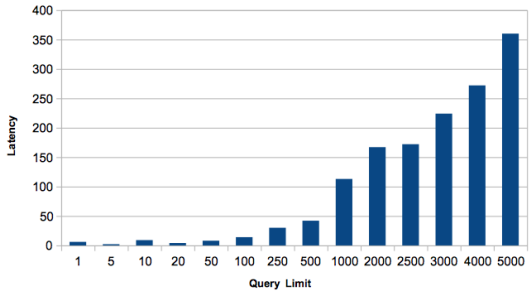


Figure 2: The query latencies when just the Patient ID is used as a filter, with active security restrictions

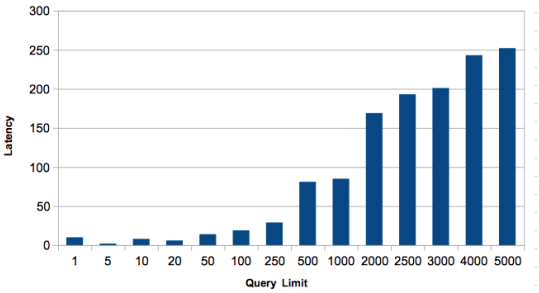


Figure 3: The query latencies when the Patient ID and the date are used as filters, without any security restrictions

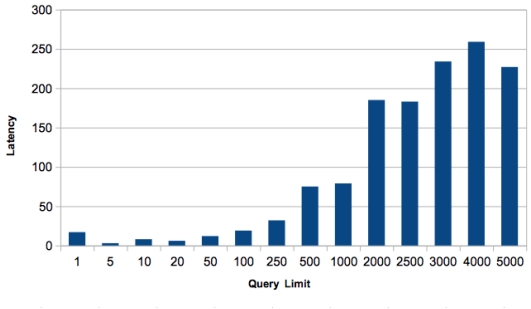


Figure 4: The query latencies when just the Patient ID and the date are used as filters, with active security restrictions

running the same select operation at first for just one document, then later for 5000. Latency is the time spent up to the server sent the response after it received the request.

These benchmarks tell us not only that despite the heavy load, our system scales at least as well as the PostgreSQL solution, or even better, but more importantly, when the security filter was active on the server there was very little difference in latency.

Conclusions

Although we still have a long road ahead of us before we have a complete, ready-to-use IT Health development stack at our disposal, it is safe to say, that our current results look most promising. Not only did we integrate a database, which is much more appropriate for the present requirements and trends of the industry than the classic, relation-based approaches, but we also created a basis for security with a fast authorization process and a design ready for multiple applications and multiple types of users. In the coming months we plan to further elaborate this framework, integrate it in our current developments, and test it with both veteran and junior developers.

Acknowledgement

This research was supported by the EU-funded Hungarian grant EFOP-3.6.1-16-2016-00008.

References

[1] *LoopBack Documentation*, Available: http://loopback.io/doc/, Accessed: 21 March 2018

[2] *Angular Architecture overview*, Available: https://angular.io/guide/architecture, Accessed: 21 March 2018

[3] *Ionic Framework Core Concepts*, Available: https://ionicframework.com/docs/intro/concepts/, Accessed: 21 March 2018

[4] Cassandra, A., *Apache cassandra*, Google Scholar, 2015.

[5] Shvachko, Konstantin, et al., *The hadoop distributed file system.*, Mass storage systems and technologies (MSST), 2010 IEEE 26th symposium on. IEEE, 2010.

[6] *FHIR Overview*, Available: https://www.hl7.org/fhir/overview.html, Accessed: 21 March 2018

[7] *Introduction to HAPI FHIR*, Available: http://hapifhir.io/docindex.html, Accessed: 21 March 2018

[8] *FHIR Security and Privacy Module*, Available: http://hl7.org/fhir/secpriv-module.html, Accessed: 21 March 2018

[9] Mandel, Joshua C., et al., *SMART on FHIR: a standards-based, interoperable apps platform for electronic health records.*, Journal of the American Medical Informatics Association 23.5, 2016, pp. 899-908.

[10] *hospitalrun.io, FHIR Implementation project*, Available: https://github.com/HospitalRun/hospitalrun-server/projects/1, Accessed: 21 March 2018

[11] McDonald, Clement J., et al., *LOINC, a universal standard for identifying laboratory observations: a 5-year update.*, Clinical chemistry 49.4, 2003, pp. 624-633.

[12] *OpenAPI Specification*, Available: https://github.com/OAI/OpenAPI-Specification/blob/master/versions/2.0.md, Accessed: 21 March 2018

[13] *FHIRbase Overview*, Available: http://fhirbase.github.io/docs.html, Accessed: 21 March 2018

Ant Colony Optimization Based Algorithm For Solving Scheduling Problems with Setup Times on Parallel Machines

Zsolt Mihály, Zsombor Sentes, Zoltán Lelkes

Abstract: In this paper, a production scheduling problem with sequence-dependent setup times on a set of unrelated parallel machines is addressed. The objective function is to minimize the total setup time . An algorithm based on ant colony optimization combined with a heuristic is proposed for solving large problems efficiently. It is shown that even a simpler version of the problem can not be tackled with MILP. ACO gives good results for the simpler problem version in a reasonable time. Even ACO can not give good results for the industrial problem. However, ACO combined with the heuristic can give us satisfactory results for the industrial problem in a reasonable time.

Keywords: production scheduling, ant colony optimization, heuristic, setup time, parallel machines, industry

Introduction

Production scheduling is a key success factor in all manufacturing industries. The choice of the schedule has a powerful impact on the cost of production [7]. Advanced optimization algorithms are needed in order to solve production scheduling problems (PSPs) efficiently.

Total setup time (TST) has a crucial effect on the profitability of equipement. In the latest review of scheduling problems with setup times, there are only two articles addressing TST when parallel machines are considered [1]. Cevikcan et al. developed an expert system [4]. Dinh et al. tackled the problem using MILP [6].

ACO is a widely used technique for solving PSPs with sequence dependent setup times. E. g. Arnaout et al. used ACO in order to minimize the makespan on unrelated parallel machines with sequence-dependent setup times [3], [2].

In this paper, a PSP with sequence-dependent setup times on a set of unrelated parallel machines is addressed. The objective function is to minimize total setup time. An algorithm based on the combination of ant colony optimization and a heuristic is proposed for solving large problems efficiently. To our best knowledge, this is the first paper which addresses the described problem using ACO.

The rest of the paper is organized as follows. The second part describes the examined PSP. The third part outlines the proposed algorithm. The fourth part shows the computational results. Finally in the fifth part, a concise conclusion about the research is presented.

Problem Statement

In this PSP, a set of n independent jobs should be scheduled on m unrelated parallel machines. Only one job can be executed on a machine at the same time. Each job j has a processing time on each machine m $p_{j,m}$. Sequence and machine dependent setup times G_{m,j_1,j_2} are also considered in this problem. G_{m,j_1,j_2} is the setup time when machine m switches the production from job j_1 to job j_2. The objective is to minimize total setup time. The examined PSP can be described as a mixed integer linear programming (MILP) model. A simplified version of the problem is already NP-hard [9]. It follows that the examined problem is NP-hard, too.

Proposed algorithm

The proposed algorithm combines ACO and a heuristic. The algorithm starts from an initial schedule. After the initialization of the algorithm, a solution is constructed for each ant. The best solutions are improved using the heuristic. The heuristic tries to find groups for ungrouped jobs. Groups consist of adjacent jobs without setup times. The heuristic checks for each ungrouped task if it can be placed into a group without impairing the objective. In the next step, the pheromones are updated. The convergence factor is calculated based on the new pheromone values. If the stop criterion is met, the algorithm stops.

The stop criterion is that the convergence factor is higher than a certain value. If the stop criterion is not met, the algorithm continues with constructing new solutions. A simplified flow chart of the algorithm can be seen on Figure 1.

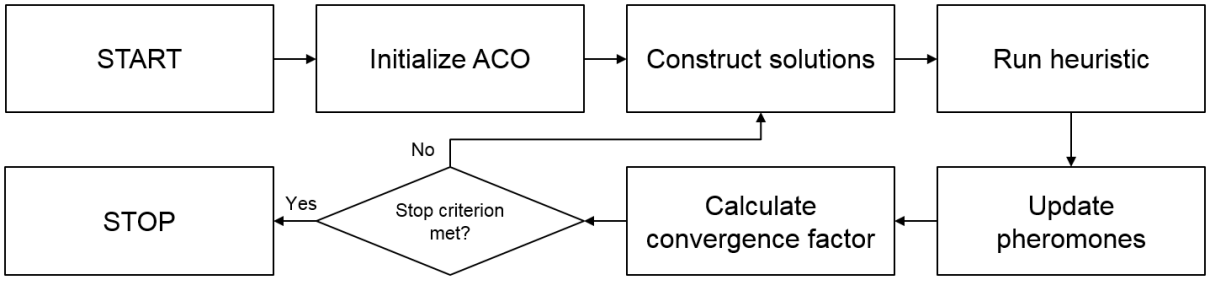


Figure 1: The outline of the proposed algorithm

The algorithm has been implemented in AIMMS modelling language [8]. It has already been used in other studies with success. E. g. Czuczai et al. used it for solving complex scheduling problems with rolling operation algorithm [5].

Computational results

Comparison with MILP

ACO has been compared with MILP on a simpler version of the investigated problem. There is only one machine to schedule. The objective is to minimize makespan. Both algorithms have been run on seven scheduling problems. The problems are different in the number of jobs: 10, 15, 20, 25, 50, 100, 150. When MILP could find optimal solution then ACO could find the same optimal solution as well. When the number of jobs is 25, MILP runs almost 40 times longer. When the number of jobs is at least 50, MILP can not find the optimal solution in a reasonable time. Even smaller instances of the simplified industrial problem can not be solved by MILP. On the other hand, ACO can give good results in a reasonable time. The data regarding the optimal solutions can be seen on Table 1. The runtime data regarding the investigation can be seen on Figure 2.

Table 1: The optimal makespans found by MILP and ACO

Number of jobs	10	15	20	25	50	100	150
MILP	787	1071	1355	1696	No data	No data	No data
ACO	787	1071	1355	1696	3290	6529	9881

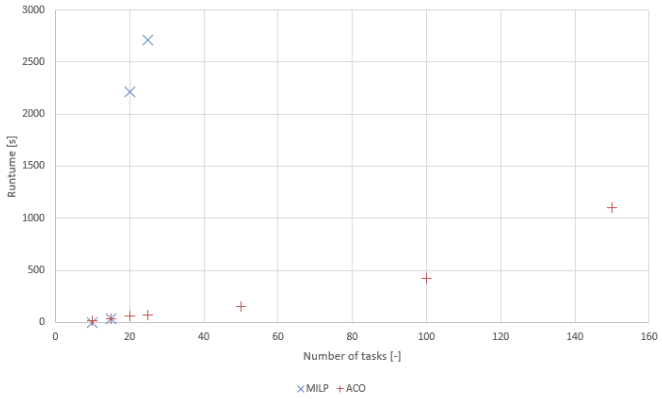


Figure 2: Comparison of MILP and ACO runtimes on single machine

Examining the proposed algorithm on the stated problem

Even ACO does not give satisfactory results when the industrial problem needs to be solved. It can happen that the new schedule is not better then the initial. For this reason, ACO has been combined with a heuristic. The ACO combined with the heuristic is able to give satisfactory results. In this part, a large problem is investigated. 193 jobs need to be scheduled on 10 machines.

The algorithm starts from an initial schedule. The initial schedule has been created by a real-life factory using a greedy randomized adaptive search procedure. The optimized schedule is shorter by 311 minutes, that is by 2.2% than the initial schedule. The runtime was 420 seconds. ACO solution construction and the heuristic consumed the most time. 212 seconds were spent on ACO solution construction, and 91 seconds on the heuristic. The initial and the optimal schedule can be seen on Figure 3 and 4.

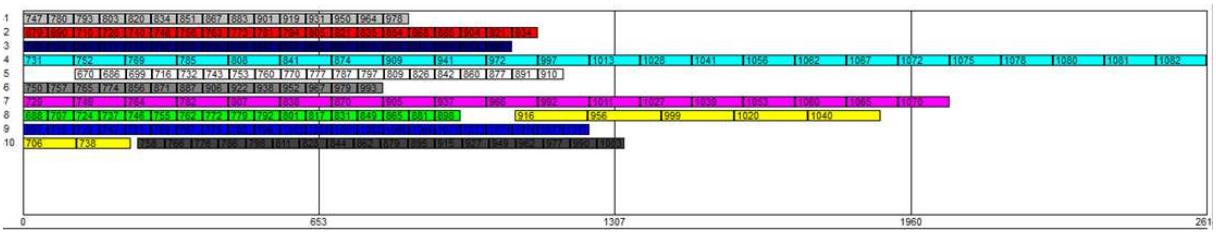


Figure 3: Initial schedule

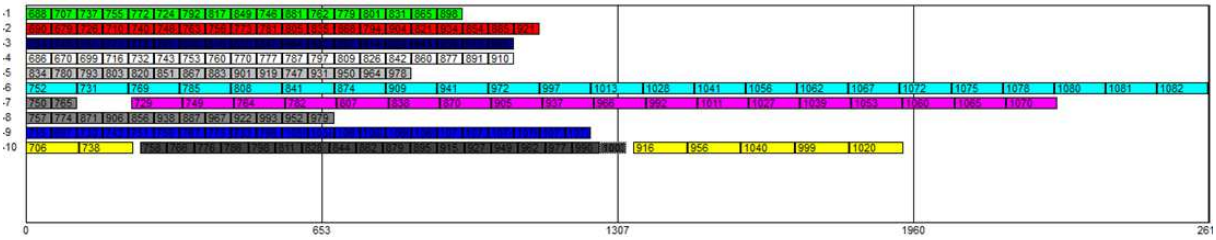


Figure 4: Optimized schedule

Conclusions

In this paper, a production scheduling problem with sequence-dependent setup times on a set of unrelated parallel machines is addressed. The objective function is to minimize total setup time. An algorithm based on the combination of ant colony optimization and a heuristic is proposed for solving large problems efficiently. It is shown that even a simpler version of the problem can not be tackled with MILP. ACO with the heuristic can give us satisfactory results in a reasonable time. A large problem has been solved with the proposed algorithm. 193 jobs needed to be scheduled on 10 machines. The optimized schedule is shorter by 311 minutes, that is by 2.2% than the initial schedule.

Acknowledgements

This research is supported by EFOP-3.6.1-16-2016-00006 "The development and enhancement of the research potential at John von Neumann University" project. The Project is supported by the Hungarian Government and co-financed by the European Social Fund.

References

[1] Allahverdi, A. The third comprehensive survey on scheduling problems with setup times/costs. *European Journal of Operational Research*, 246:345–378, 2015.

[2] Arnaout, J.-P., Musa, R., and Rabadi, G. A two-stage ant colony optimization algorithm to minimize the makespan on unrelated parallel machines-part ii: enhancements and experimentations. *Journal of Intelligent Manufacturing*, 25:43–53, 2014.

[3] Arnaout, J.-P., Rabadi, G., and Musa, R. A two-stage ant colony optimization algorithm to minimize the makespan on unrelated parallel machines with sequence-dependent setup times. *Journal of Intelligent Manufacturing*, 21:693–701, 2010.

[4] Cevikcan, E., Durmusoglu, M. B., and Baskak, M. Integrating parts design characteristics and scheduling on parallel machines. *Expert Systems with Applications*, 38:13232–13253, 2011.

[5] Czuczai, B., Farkas, T., Rev, E., and Lelkes, Z. Rolling operation algorithm for solving complex scheduling problems. *Industrial and Engineering Chemistry Research*, 48:3898–3908, 2009.

[6] Dinh, T.-C. and Bae, H. Parallel servers scheduling with dynamic sequence-dependent setup time. *Smart Innovation, Systems and Technologies*, 16:79–87, 2012.

[7] Pinedo, M.L. *Scheduling*. Springer-Verlag, Berlin, 2016.

[8] Roelofs, M. and Bisschop, J. *AIMMS The user's guide*. AIMMS B.V., AP Haarlem, The Netherlands, 2016.

[9] Webster, S.T. The complexity of scheduling job families about a common due date. *Operations Research Letters*, 20:65–74, 1997.

Instantiation context aware types in C++

Zsolt Parragi, Zoltán Porkoláb

Abstract: Programming languages usually limit the information available to types. Object oriented languages, such as C++ or Java typically define types as definition context aware, but instantiation context free: types are able to use other types and language constructs declared or defined before their definition, but they aren't aware of where, or in which context they are instantiated. While this is practical in most situations, both for considering the compilation speed and the complexity of the language, it also limits the expressiveness of the language. In this article, we present a way instanation context aware types could be implemented in modern C++, and a few examples where they could be useful.

Keywords: C++, Dependency Injection, Templates, Metaprogramming

Motivation

Types in most programming languages are aware of their definition context, but have no information about their instantiation context. This can be shown on a simple example:

```
// ...
struct T1 {
  // ...
};
struct T3;
struct T2 {
  T1 t1;
  T3* t3;
  // ...
};
// ...
struct Foo {
  T1 a;
  T2 b;
  T3 c;
};
```

In this simple program snippet, the types, such as T1 or Foo, are aware of the context of their definition. For example `Foo` is able to access the definition of `T1` as long as it is defined before it, or similarly, `T2` is able to use `T1`, which is defined before, or `T3*`, as `T3` is forward declared before its definition. Similarly it also would be able to use global variables, global functions, or member functions defined in existing types, if their access restrictions allows. This demonstrates that types in C++ are aware of their definition context.

Howewer, the language specification also states that `b`, an instantiation of `T2` in `Foo` has no knowledge about `a`, another data member defined just before it in `Foo`, or `c`, the data member defined after it, but still in the same scope. Similarly, `T1` does not know that it is instantiated in `T2` and `Foo`, or how many other data members, functions `Foo` has, or anything about their types. As for data members, the direct scope containing the type is the enclosing type, this means that types in C++ aren't aware of their instantiation context.

This limitation is intended, both for the sake of the compilers, and developers:

From a compilers perspective, it reduces the complexity of the implementation, and also introduces flexibility: It reduces the complexity, because the compiler does not have to do additional computations for each type for each instantiation – which, as instantiations are usually more numerous than definitions, would likely have a significant impact on compilation times. And while it is a limitation, by adding this restrictions, it also introduces flexibility: instantation context awareness is only possible, if the compiler evaluates the definition of everything – including member functions – in the type for the type for every instantiation. This is only possible if the compiler has the ability to generate a different intermediate or binary result each time, for which it needs access to the source each time. For this,

similarly to how templates work, the compiler would have to see the actual source code for everything related to that type – only not only for templates, but for every type. By making types instantation context free, the languange can be more flexible at other places – for example, it is able to allow "hidden" definitions for member function bodies.

From the developers perspective, added complexity isn't always better. As instantiation information is not required for most types, removing this restriction would only increase the possible errors in most programs, and would make runtime or compile time diagnostic harder.

Dependency Injection

Howewer developers sometimes do need this feature, or something similar, and compensate for its unavailability with design patterns. The most trivial example for this is when the data members of a type have some dependencies between each other. In the above example, `T2` holds a pointer to an instance of `T3`, which oftem injected as a parameter to its constructor, or as a setter. This principle, referred as dependency injection[5] is also supported by several frameworks, such as Boost DI[4].

These dependencies are also commonly represented as interfaces instead of concrete types, adding the ability to use different implementations as parameters. While this principle aims to help decoupling of the implementations, it is easy to misuse: when working with several implementations of some interfaces, sometimes the implementations rely on hidden properties of the type implementing the other interface instead of only on what is visible on the public interface. There could be many reasons for this, for example as a performance optimization, a design simplification, or simply forcing something new to work using legacy interfaces. Code relying this usually uses a `dynamic_cast`, or sometimes simply a `static_cast` on the interface in its implementation, and results in failures when invoking with an incorrect parameter.

Even when this isn't the case, an user of an interface could add additional requirements to the dependencies it accepts: for example on the surface it could accept a generic container, but a commend and a runtime check in the constructor would check that the given container only holds at most `N` items – otherwise the constructor would throw, preventing the creation of the object.

This is a neccessary side effect of using dependency injection: this technique allows the use of a runtime configuration when initializing a software product, theoretically starting up the program in several possible combinations. With this generalization, the developers either have to either check the validity of every possible combinations during the compilation of the program, or have to fall back on initialization time error messeges. As the latter is easier to do, and still results in an early error detection, it is the oblivious choice.

In practice however dependency injection is often used as an internal helper for development instead of a configuration option for end users: the dependencies are hard coded into the software, and the only goal of the DI framework is to produce a cleaner, better structured code, with less hand written boilerplate. In this case, the actually used configurations are known at compilation time, still, part of the error checks are deferred until the actual execution of the program. Even worse, these `if`s and `throw`s can be replaced with `assert`s, as developers assume that the code is throughly tested in a debug build, and want to increase the runtime performance of the released program.

This is where instantiation context aware types could help: if the required checks could be made during compilation time with the specific types known, developers could change runtime checks only present in debug builds into compile time checks, resulting in earlier and consistent problem detection.

Memory efficient dependencies

Additionally, if the types have enough information about their instatiation context, and the other types in that context, they could deduce the address of the parent type and other data members without storing actual pointers to them. This would result in a technique which not only allows additional compile time checks for the used types, but also presents a way for a memory-efficient dependency injection system, or rather, in the use of the composition pattern[1] as the dependency injection framework. While memory efficiency is not that important in perconal computers or handheld devices with the amount of memory available, in microcontrollers with only a few kilobytes of available memory it is still a primary question.

With context aware types, writing memory efficient, but also generic and safe types becomes a possobility, which could help embedded developers in writing better code. A possible use of this technique, which could be useful even on desktop systems is a memory efficient obvervable type: a type where every data member could be independently observed, without adding additional data members to every member type - and thus, increasing their size. Instead all data members could store their obververs in a common list, while maintaining the illusions that the user code directly attaches obsevrvers to the members.

```
struct StructWithObservableMembers {
  Observable<int> a;
  Observable<bool> b;
  Observable<floant> c;
private:
  ListOfObserversForEveryMember observers;
};
// ....
StructWithObservableMembers s;
s.a.observe([](...){ ... });
```

Conclusion

We have shown two examples where context aware types could result in definite advantates, with earlier validation, or with increasing the memory efficiency of the program. While this technique is not supported out of the box by traditional object oriented languages, the features provided by modern C++ standards allow a possible implementation with the use of templates. We developed a prototype framewrok using either the C++17 standard[2] with some commonly available compiler extensions, or with features currently present in the C++20 working draft[3]. While our prototype requires changes in the types which have to be instantation context aware, and also for the types that contain instantation context aware types, it also handles both of the examples mentioned previously, and could provide a desing alternative in some situations.

References

[1] Erich Gamma et al. *Design Patterns: Elements of Reusable Object-oriented Software* . Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1995. isbn : 0-201-63361-2.

[2] ISO *ISO/IEC 14882:2017 Information technology - Programming languages - C++.* In: Geneva, Switzerland: International Organization for Standardization, 2017.

[3] ISO *Working Draft, Standard for Programming Language C++.* 2018. url: http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2018/n4727.pdf (visited on 03/31/2018)

[4] Krzysztof Jusiak. *[Boost].DI.* 2018. url: http://boost-experimental.github.io/di/ (visited on 03/31/2018)

[5] Dhanji R. Prasanna. *Dependency Injection*. 1st. Greenwich, CT, USA: Manning Publications Co., 2009. ISBN: 193398855X, 9781933988559

LZ based compression benchmark on PE files

Zsombor Paróczi

Abstract: The key element in runtime compression is the compression algorithm, that is used during processing. It has to be small in enough in decompression bytecode size to fit in the final executable, yet have to provide the best compression ratio. In our work we benchmark the top LZ based compression methods on Windows PE files (both exe and dll files), and present the results including the decompression overhead and the compression rates.

Keywords: lz based compression, compression benchmark, PE benchmark

Introduction

During runtime executable compression an already compiled executable is modified in ways, that it still retains the ability to execute, yet the transformation produces smaller file size. The transformations usually exists from multiple steps, changing the structure of the executable by removing unused bytes, adding a compression layer or modifying the code in itself. During the code modifications the actual bytecode can change, or remain the same depending on the modification itself.

In the world of x86 (or even x86-64) PE compression there are only a few benchmarks, since the ever growing storage capacity makes this field less important. Yet in new fields, like IOT and wearable electronics every application uses some kind of compression, Android apk-s are always compressed by a simple gzip compression. There are two mayor benchmarks for PE compression available today, the Maximum Compression benchmark collection [1] includes two PE files, one DLL and one EXE, and the Pe Compression Test [2] has four exe files. We will use the exe files during our benchmark, referred as small corpus. More detailed results we have a self-collected corpus of 200 PE files, referred as large corpus.

When approaching a new way to create executable compression, one should consider three main factors. The first is the actual compression rate of the algorithms, since it will have the biggest effect on larger files. The second is the overhead in terms of extra bytecode within the executable, since the decompression algorithm have to be included in the newly generated file, using large pre-generated dictionary is not an option. This is especially important for small (< 100kb) executables. The third one has the lowest priority, but still important: the decompression speed. The decompression method should not require a lot of time to run, even on a resource limited machine. This eliminates whole families of compressions, like neural network based (PAQ family) compressions.

Split-stream methods are well know in the executable compression world, these algorithms take advantage of the structural information of the bytecode itself, separating the opcode from all the modification flags. We used a reference implementation from the packer, kkrunchy [5].

LZ based compression methods

Compression method	Version	Source
aPlib	1.1.1	http://ibsensoftware.com/products_aPLib.html
Lzlib	1.10	https://www.nongnu.org/lzip/lzlib.html
LZMA	9.35	https://www.7-zip.org/sdk.html
Zopfli	2017-07-07	https://github.com/google/zopfli
Zstandard	1.3.3	https://facebook.github.io/zstd/
CSC	2016-10-13	https://github.com/fusiyuan2010/CSC
Brotli	1.0.3	https://github.com/google/brotli

Table 1: Libraries used during the benchmark

LZ based compression methods (LZ77/LZSS/LZMA families) are well fitted for this compression task, since they usually have relatively small memory requirement (< 64 Mb), they use Lempel-Ziv compression methods [3] and maybe some Huffman tables or hidden Markov model based approaches. These methods also result in simple algorithms, resulting in small size in terms of decompression byte-code. During the last few years there are a lot of new LZ based compression methods, the mayor ones are Zstandard (zstd) from Facebook and Zopfli from Google. The selected libraries can be seen on Table 1., these are the top LZ familly libraries for generic purpose compression regarding to an extensive LZ benchmark [4].

Benchmark

During the benchmark we constructed a system, which is capable of extracting different sections from the executables, apply split-stream and a compression on it to create a well detailed benchmark result. During the benchmark we run each compression method on each section, then run each compression method with split-stream on executable sections.

Results: compression ratio

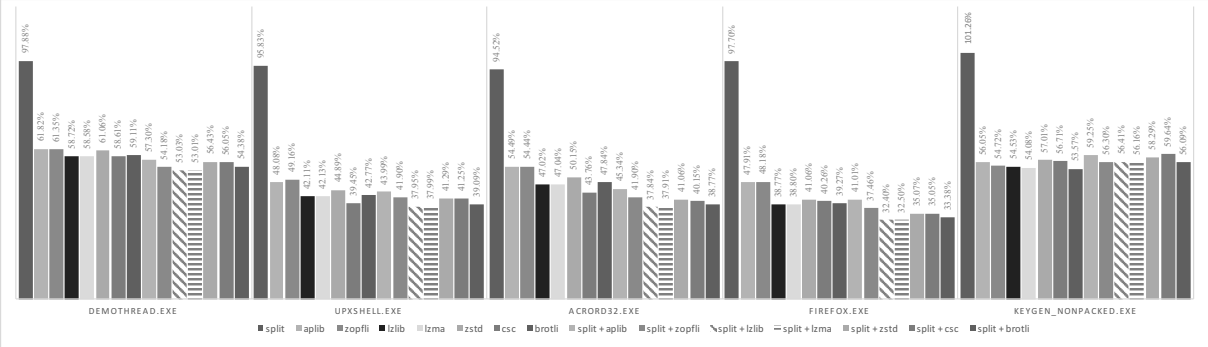


Figure 1: Resulting section size compered to the original on the small corpus files

The detailed results for each test case on the small corpus can be seen on Figure 1. As you can see applying split-stream before the compression is useful in most of the cases (except for the smallest executable, which suffered from the overhead of this method - splitting 1 byte instructions into base instruction + mod flags). The rates for each compression varies between test cases, but Lzlib, LZMA, Brotli are clearly the best for the small corpus, followed by zstd, CSC, Zopfli and aPlib. There is a constant improvement when using split-stream. Only for really small executable aPlib is the best, due to the simplicity of the algorithm itself. All of these results were verified during our large corpus benchmark.

aplib	lzma	s + aplib	s + zopfli	s + lzlib	s + lzma	s + zstd	s + csc	s + brotli
47%	42%	44%	41%	40%	39%	42%	42%	40%

Table 2: Average compression rates on code section

aplib	zopfli	lzlib	lzma	zstd	csc	brotli
40%	37%	35%	34%	38%	35%	33%

Table 3: Average compression rates on non-code section

The actual compression rates on the large corpus can be seen on Table 2. and 3. (split-stream is annotated as s). As you can see the ratio between each compression rate on average is really small, for code sections split-stream really helps. For code section LZMA, Lzlib and Brotli are the best, followed by Zopfli and CSC. For non-code section we had a larger variety of results, since the non-code sections can contain any datatype. Since it has a more loose structure and less density, the compression rates are higher. It is interesting, that Brotli was the winner in these tests, but as it turned out Brotli has a large dictionary prebuilt into the algorithm, that helps with compressing text. LZMA, LZlib, CSC produced just 1-2% lower rates, followed by zstd and Zopfli. Obviously aPlib was the worst in both tests, since it contains the most simple algorithm for compression. Since the PE sections tend to be less then 3 Mb, the larger the section the more compression rate we can achieve but to the lower entropy.

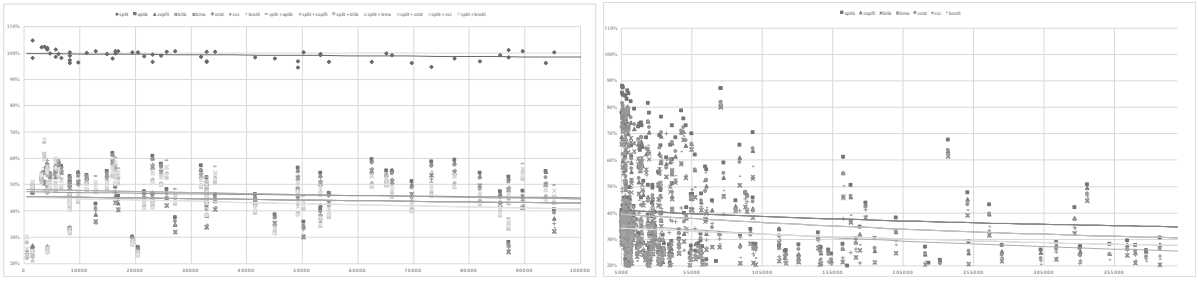


Figure 2: Compression rates vs file size: left the code section, right the non-code section results

Results: final file size with decompression bytecode

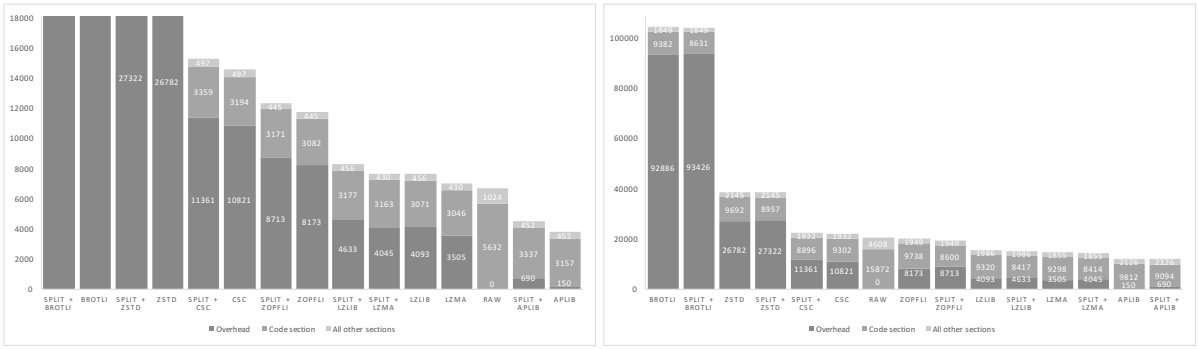


Figure 3: Final executable size: keygen_unpacked.exe, DemoThread.exe

Since the decompression code has to be included in the final executable, we also benchmarked how the decompression overhead code effects the final file size. As you can see on Figure 3. for smaller executables the overhead is what really defines the final result. All of the decompression methods were packed with aPlib, since aPlib has a decompression code size of 150 bytes, and above 1.000 bytes it is better to compress the decompression code with aPlib. Some of the more complex methods (namely zstd, Brotli, CSC) has relatively large data tables in the decompression code. Same goes for the split-stream code, which is above 1kByte uncompressed, and 540 byte compressed with aPlib.

Our final results suggest, that there is no "golden" LZ based compression with split-stream method for all the executables. We consider 3 categories based on the executable size: for small files (< 50kB) size aPlib is the clear winner with 150 byte decompression code, maybe with split-stream if the executable section is large. For medium size (< 500 kB) split-stream with aPlib or split-stream with LZMA (aPlib compressed) should be used. For larger files split-stream with LZMA (aPlib compressed) or split-stream with Lzlib (aPlib compressed) should be used. You can see the the best performing algorithm on Figure 5. for the large corpus. For some special cases each combination can be the winner in regards of final compression size. CSC (without split-stream), Lzlib (without split-stream) and LZMA (without split-stream) can outperform the others in some cases.

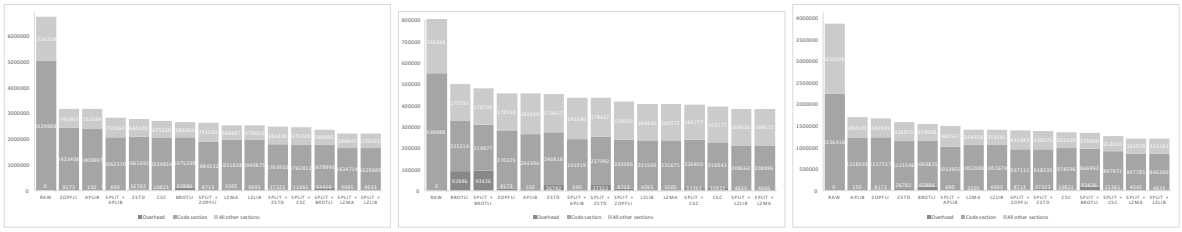


Figure 4: Final executable size: Firefox.exe, UPXShell.exe, acrord32.exe

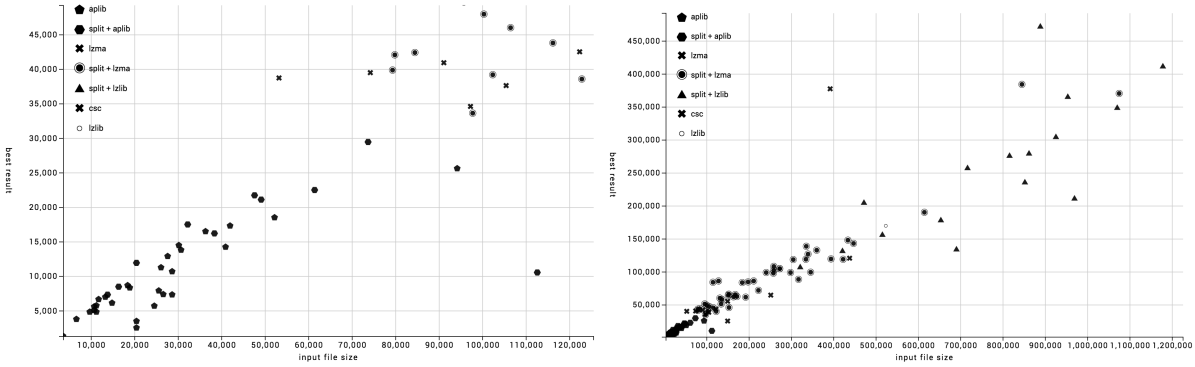


Figure 5: Raw and compressed file size using the best method

References

[1] Lossless data compression software benchmarks / comparisons, *https://www.maximumcompression.com/* (Visited 2018 2018-03-04).

[2] Pe Compression test by Ernani Weber *http://pect.atspace.com/* (Visited 2018-03-04).

[3] Ziv J., Lempel A., A universal algorithm for sequential data compression, *IEEE Trans. on Inf. Th. IT-23* (1977) 337-343.

[4] LZbench *https://github.com/inikep/lzbench/* (Visited 2018-03-04).

[5] Fabian Giesen. Working with compression. *Breakpoint conference* (2006)

List of Authors

Aljawabrah, Nadera: University of Szeged, Hungary,
E-mail: nadera@inf.u-szeged.hu

Bácsi, Sándor: Budapest University of Technology and Economics, Hungary,
E-mail: bacsi.sandor93@gmail.com

Bálint, Csaba: Eötvös Loránd University, Hungary,
E-mail: csabix.balint@gmail.com

Bartha, Dénes: Eötvös Loránd University, Hungary,
E-mail: denesb@gmail.com

Belákovics, Ádám: Budapest University of Technology and Economics, Hungary,
E-mail: beladi96@gmail.com

Bouafia, Khawla: Eötvös Loránd University, Hungary,
E-mail: bouafiakhawla24@gmail.com

Brunner, Tibor: Eötvös Loránd University, Hungary,
E-mail: bruntib@caesar.elte.hu

Budai, Ádám: Budapest University of Technology and Economics, Hungary,
E-mail: budai.adam@aut.bme.hu

Czémán, Arnold: Budapest University of Technology and Economics, Hungary,
E-mail: czeman.arnold@cloud.bme.hu

Csákvári, Máté: Eötvös Loránd University, Hungary,
E-mail: csarli1121@gmail.com

Domonkos, Sándor Balázs: University of Szeged, Hungary,
E-mail: sada1988@gmail.com

Gál, Péter: University of Szeged, Hungary,
E-mail: galpeter@inf.u-szeged.hu

Gelle, Kitti: University of Szeged, Hungary,
E-mail: kgelle@inf.u-szeged.hu

Homolya, Viktor: Universitiy of Szeged, Hungary,
E-mail: viktor.homolya@gmail.com

Horváth, Gábor: Eötvös Loránd University, Hungary,
E-mail: xazax.hun@gmail.com

Hudoba, Péter: Eötvös Loránd University, Hungary,
E-mail: hudi1989@gmail.com

Hussain, Abrar: University of Szeged, Hungary,
E-mail: Abrarlaghari@hotmail.com

Ilyés, Enikő: Eötvös Loránd University, Hungary,
E-mail: ilyese@inf.elte.hu

Jánki, Zoltán Richárd: University of Szeged, Hungary,
E-mail: jankiz@inf.u-szeged.hu

Kalmár, György: University of Szeged, Hungary,
E-mail: kalmargy@inf.u-szeged.hu

Kicsi, András: University of Szeged, Hungary,
E-mail: akicsi@inf.u-szeged.hu

Kovács, Tibor: Budapest University of Technology and Economics, Hungary,
E-mail: kovacs.tibor@outlook.hu

Lam Van Thanh, Binh: Budapest University of Technology and Economics, Hungary,
E-mail: lvtbinh@hit.bme.hu

Lékó, Gábor: University of Szeged, Hungary,
E-mail: leko@inf.u-szeged.hu

Li, Yangyuan: Budapest University of Technology and Economics, Hungary,
E-mail: wlqsb@hotmail.com

Lukács, Dániel: Eötvös Loránd University, Hungary,
E-mail: dlukacs@caesar.elte.hu

Luksa, Norbert: Eötvös Loránd University, Hungary,
E-mail: luksan@eotvos.elte.hu

Márkus, András: University of Szeged, Hungary,
E-mail: markus.andras@stud.u-szeged.hu

Mattyasovszky-Philipp, Dóra: Eötvös Loránd University, Hungary,
E-mail: philippd@caesar.elte.hu

Mester, Abigél: University of Szeged, Hungary,
E-mail: mester@inf.u-szeged.hu

Mihály, Zsolt: University John von Neumann, Hungary,
E-mail: mihaly.zsolt@gamf.uni-neumann.hu

Mishra, Biswajeeban: University of Szeged, Hungary,
E-mail: mishra@inf.u-szeged.hu

Nagy, Dávid: Debreceni Egyetem, Hungary,
E-mail: nagy.david@inf.unideb.hu

Oláh, Norbert: University of Debrecen, Hungary,
E-mail: olah.norbert@inf.unideb.hu

Pap, Gergely: University of Szeged, Hungary,
E-mail: papg@inf.u-szeged.hu

Paróczi, Zsombor: Budapest University of Technology and Economics, Hungary,
E-mail: paroczi@tmit.bme.hu

Parragi, Zsolt: Eötvös Loránd University, Hungary,
E-mail: zsoltparragi@caesar.elte.hu

Pengő, Edit: University of Szeged, Hungary,
E-mail: pengoe@inf.u-szeged.hu

Pusztai, László: University of Debrecen, Hungary,
E-mail: pusztai.laszlo@inf.unideb.hu

Rill, Róbert Adrián: Eötvös Loránd University, Hungary,
E-mail: rillroberto88@yahoo.com

Ságodi, Zoltán: University of Szeged, Hungary,
E-mail: sagodiz@inf.u-szeged.hu

Selmeci, Attila: Óbuda University, Hungary,
E-mail: selmeci.attila@amk.uni-obuda.hu

Szabó, Zoltán: University of Szeged, Hungary,
E-mail: szaboz@inf.u-szeged.hu

Szécsi, Péter: Eötvös Loránd University, Hungary,
E-mail: Peterszecsi95@gmail.com

Szekér, Szabolcs: University of Pannonia, Hungary,
E-mail: szeker@dcs.uni-pannon.hu

Szűcs, Judit: University of Szeged, Hungary,
E-mail: jszucs@inf.u-szeged.hu

Tamás, Judit: University of Miskolc, Hungary,
E-mail: tamasjudit@iit.uni-miskolc.hu

Tóth, László: University of Szeged, Hungary,
E-mail: premissa@inf.u-szeged.hu

Tóth, Gabriella: Eötvös Loránd University, Hungary,
E-mail: kistoth@inf.elte.hu

Tóth, Ákos: University of Debrecen, Hungary,
E-mail: toth.akos@inf.unideb.hu

Varga, Viktor: Eötvös Loránd University, Hungary,
E-mail: vvarga90@gmail.com

Véges, Márton: Eötvös Loránd University, Hungary,
E-mail: vegesm@gmail.com

Verma, Chaman: Eötvös Loránd University, Hungary,
E-mail: phd@inf.elte.hu

Notes

Supported by the project *"Integrated program for training new generation of scientists in the fields of computer science"*, № **EFOP-3.6.3-VEKOP-16-2017-00002**. The project has been supported by the European Union and co-funded by the European Social Fund.