Global Optimization, Directed Acyclic Graphs (DAG), and the COCONUT Environment

University of Vienna
H. Schichl

Overview

- Basic definitions
- Reasons for doing research in global optimization
- Problem and Real-World Examples
  - Constraint Satisfaction Problems (CSP)
  - Global Optimization (GOP)
- Directed Acyclic Graphs (DAG)
- Methods
  - The Branch-and-Bound Principle
  - Constraint Propagation
  - Relaxation
- The COCONUT Environment
  - Design and Implementation
- Open Problems

Basic definitions

- **Constraint satisfaction problem (CSP):**
  - Take finitely many **variables** and **domains** for their values (a subset of the reals, integers, or more abstract sets),
  - Impose finitely many **constraints** on these variables (e.g., must be different, their sum is at most 10, their product is positive, and the like),
  - Find one or all **feasible points**, i.e. a set of values for all variables from their respective domains such that all constraints are satisfied.
- **Global optimization problem (GOP):**
  - Take a constraint satisfaction problem and add a real valued function, the **objective function**, assigning a value to all feasible points.
  - Find those feasible points, whose objective function value is minimal (maximal) among all feasible points.

Reasons for researching global optimization

- **Earn money:**
  - Global optimization is **NP-hard**.
  - Many constraint satisfaction problems are **NP-complete**.
  - Getting a deep understanding of GOP and CSP may provide insight in the **P vs NP** problem.
  - Solving the P vs NP problem will get you **one million dollars**.
- **If you don't manage to solve the P vs NP problem:**
  - **Protein folding** is a huge global optimization problem (money from the pharmaceutical industry).
  - **Robotics** is a fast developing industrial area, where a lot of CSPs and GOPs have to be solved efficiently.
  - Many **chemical engineering** tasks lead to global optimization problems. Solving them properly is a prerequisite to the design of more efficient chemical reactors.

Reasons for researching global optimization (ctd.)

- Industrial problems leading to GOP:
  - Design of mechanical structures leads to global optimization problems when it comes to safety related issues (aircrafts, bridges, buildings).
  - Structural integrity of mechanical products (motors, breaks,...) leads to global optimization problems coupled with ordinary or partial differential equations.
- Other industrial applications (traveling salesman, logistics, commodity flow problems, scheduling,...) also lead to GOPs.
- However, there is something special with the blue problems in the list above:
  The solution to those is useful only if the global optimum can be found with guarantee!
  Thus, simulated annealing, genetic algorithms, and the like cannot be used!

Reasons for researching global optimization (ctd.)

- Global optimization is everywhere.
- Living your life is a global optimization problem.
- The constraints are apparent most of the time, though some 'hidden' ones might exist.
- Finding the correct objective function might prove difficult, and might be more of a philosophical problem rather than a purely mathematical one.
- At the end, you might realize that just earning money is probably not the best way to live your life.

Real World Problem

He pulled open the next door ... there was nothing frightening in here, just a table with seven differently shaped bottles standing on it in a line.

“Snape's,” said Harry. “What do we have to do?”

They stepped over the threshold and immediately a fire sprang up behind them in the doorway... At the same instant, black flames shot up in the doorway leading onwards. They were trapped.

“Look!” Hermione seized a roll of paper lying next to the bottles. Harry looked over her shoulder to read it:

Real World Problem (ctd.)

Danger lies before you, while safety lies behind,
Two of us will help you, whichever you would find,
One among us seven will let you move ahead,
Another will transport the drinker back instead,
Two among our number hold only nettle wine,
Three of us are killers, waiting hidden in line,
Choose, unless you wish to stay here for evermore,
To help you in your choice, we give you these clues four:
First, however slyly the poison tries to hide
You will always find some on nettle wine's left side;
Second, different are those who stand at either end,
But if you would move onwards, neither is your friend;
Third, as you see clearly, all are different size,
Neither dwarf nor giant holds death in their insides;
Fourth, the second left and the second on the right
Are twins once you taste them, though different at first sight.

Real World Problem (ctd.)

Hermione read the paper several times... At last, she clapped her hands.

“Got it,” she said. “The smallest bottle will get us through the black fire - towards the Stone.”

...

“Which one will get you back through the purple flames?”

Hermione pointed at the rounded bottle at the right end of the line.

Modeling

- From the Latin word "modellus", typical human way of coping with the reality
- Mechanical models
  - Astronomy
  - Architecture
- Mathematical models used since Hellenic Age
  - Geometry (~600 BC (Thales) — ~300 BC (Euclid))
  - Circumference / Diameter of Earth (~250 BC (Erathostenes))
  - Arithmetic (~240 AD (Diophantus) — ~800 AD (Al-Khwarizmi))
  - Astronomy (~150 AD (Ptolemy))
  - Physics
  - Economics
- Visual models
  - Anatomy

Modeling (ctd.)

- A model is a simplified version of something that is real.
- Represents a real-world object by other, simpler objects.
- Structures the knowledge about the real world.
- Reduction to those phenomena and aspects which are considered important.
- Usefulness is restricted to its scope of application.
- Serve many different needs — the modeling goal
  - explain phenomena, make predictions,
  - control environment, decision making,
  - maintenance, normative components,
  - communication

Mathematical Modeling

- Represents a real-world problem by mathematical objects in a formalized mathematical language.
- Can be analyzed by mathematical theory and algorithms.
- Represents the knowledge about the real-world by concepts, variables, relations, data.
- Makes models accessible to computers.
  - First computers were human (usually lowly paid women).
  - Since ENIAC (1945) larger and more complicated mathematical models have become accessible.
- Models became attractive to military and industry.

Real World Problem (revisited)

Let $I = \{1, \ldots, 7\}$ and $x_i \in \{B, F, P, W\}$ for $i \in I$.

1. $\exists! i \in I : x_i = F$,
2. $\exists! i \in I : x_i = B$,
3. $|\{i \in I \mid x_i = W\}| = 2$,
4. $\forall i \in I \setminus \{1\} : (x_i = W \implies x_{i-1} = P)$,
5. $x_1 \neq W$,
6. $x_1 \neq x_7$,
7. $x_1 \neq F$,
8. $x_7 \neq F$,
9. $x_2 = x_6$,
10. $\forall i \in I : (S(i) \implies x_i \neq P)$,
11. $\forall i \in I : (L(i) \implies x_i \neq P)$,
12. $\exists! i \in I : S(i)$,
13. $\exists! i \in I : L(i)$,
14. $\forall i \in I : \neg(S(i) \wedge L(i))$

A constraint satisfaction problem

Real World Problem (again)

$$
\begin{array}{l}
x_2 = F \overset{9}{\Longrightarrow} x_6 = F \overset{1}{\Longrightarrow} \lightning \\
x_2 = B \overset{9}{\Longrightarrow} x_6 = B \overset{2}{\Longrightarrow} \lightning \\
x_2 = P \overset{9}{\Longrightarrow} x_6 = P \\
\quad | x_7 = F \overset{8}{\Longrightarrow} \lightning \\
\quad | x_7 = B \overset{5,6,7}{\Longrightarrow} x_1 \neq B \wedge x_1 \neq W \wedge x_1 \neq F \implies x_1 = P \overset{3,4}{\Longrightarrow} \lightning \\
\quad | x_7 = P \overset{3,4}{\Longrightarrow} \lightning \\
\quad \implies x_7 = W \overset{5,7}{\Longrightarrow} x_1 \neq F \wedge x_1 \neq W \\
\quad | x_1 = P \overset{3,4}{\Longrightarrow} x_3 = W \overset{1,2}{\Longrightarrow} x_4 = F \vee x_4 = B \\
\quad | \quad | x_4 = B \overset{1}{\Longrightarrow} x_5 = F \ \checkmark \ \mathbf{P\ P\ W\ B\ F\ P\ W} \\
\quad | \quad | x_4 = F \overset{2}{\Longrightarrow} x_5 = B \ \checkmark \ \mathbf{P\ P\ W\ F\ B\ P\ W} \\
\quad | x_1 = B \overset{2}{\Longrightarrow} x_3 \neq B \\
\quad | \quad | x_3 = F \overset{1,2,4}{\Longrightarrow} x_4 = P \overset{3}{\Longrightarrow} x_5 = W \ \checkmark \ \mathbf{B\ P\ F\ P\ W\ P\ W} \\
\quad | \quad | x_3 = W \overset{2,3}{\Longrightarrow} x_4 \neq W \wedge x_4 \neq B \\
\quad | \quad | \quad | x_4 = F \overset{1,2,3}{\Longrightarrow} x_5 = P \ \checkmark \ \mathbf{B\ P\ W\ F\ P\ P\ W} \\
\quad | \quad | \quad | x_4 = P \overset{1,2,3}{\Longrightarrow} x_5 = F \ \checkmark \ \mathbf{B\ P\ W\ P\ F\ P\ W} \\
\quad | \quad | x_3 = P \overset{3,4}{\Longrightarrow} x_4 = W \overset{1}{\Longrightarrow} x_5 = F \ \checkmark \ \mathbf{B\ P\ P\ W\ F\ P\ W} \\
x_2 = W \overset{9}{\Longrightarrow} x_6 = W \overset{4}{\Longrightarrow} x_1 = P \wedge x_5 = P \overset{3,6,8}{\Longrightarrow} x_7 = B \\
\quad \overset{1,2,3}{\Longrightarrow} x_3 \in \{F, P\} \\
\quad | x_3 = F \overset{1,2,3}{\Longrightarrow} x_4 = P \ \checkmark \ \mathbf{P\ W\ F\ P\ P\ W\ B} \\
\quad | x_3 = P \overset{1}{\Longrightarrow} x_4 = F \ \checkmark \ \mathbf{P\ W\ P\ F\ P\ W\ B}
\end{array}
$$

We see that the information provided in the book is not enough to solve the problem.

The size information, which Harry and Hermione have is vital for solving the CSP.

Real World Problem (again, ctd.)

- **Did J.K.Rowling know what she did when inventing that riddle?**
- **Is there a possibility of assigning bottle sizes in such a way that the riddle is solvable by Hermione?**
- **Mathematical model:**

Let $I = \{1, \ldots, 7\}$ and $y_i \in \{1, \ldots, 7\}$ for $i \in I$.

1. all_different$(y_1, \ldots, y_7)$,
2. $\forall i \in I : (y_i = 1 \iff S(i))$,
3. $\forall i \in I : (y_i = 7 \iff L(i))$,
4. $|\text{solution\_set}(HP(L, S))| = 1$,

where *HP(L,S)* denotes the CSP we have seen before with given assignment of “largest” and “smallest”.

Real World Problem (again, ctd.)

- It turns out that there are indeed many solutions to that problem.
- Eight of the solutions are essentially different in placing the smallest and largest bottles. All other bottle sizes can then be assigned arbitrarily.
- Four of these eight essentially different solutions correspond to the book.
- These four solutions are grouped in two pairs, the difference being just the assignment of the largest bottle.

$S(3), L(2) \vee L(6), x_7 = B, x_3 = F$	P W F P P W B
$S(4), L(2) \vee L(6), x_7 = B, x_4 = F$	P W P F P W B

Minesweeper

- Very well known game, for which even world championships are organized and world record lists are maintained.
- Mathematical Questions:
  - is a given game solvable by applying logical rules only?
  - Given a board preset with mines and numbers. Is there a consistent way to write numbers and mine symbols on all closed fields?

Minesweeper (2)

- **The consistency question can be modeled as a mathematical problem:**

$$\mathcal{C} \subset \mathcal{F} \qquad \mathcal{F} = \{1,\ldots,N\} \times \{1,\ldots,M\}$$

$$\sum_{k=-1}^{1} \sum_{\ell=-1}^{1} x_{i+k\; i+\ell} = c_{ij} \qquad \text{für alle } (i,j) \in \mathcal{F} \setminus \mathcal{C}$$

$$x_{ij} = 0 \qquad \text{für alle } (i,j) \in \mathcal{F} \setminus \mathcal{C}$$

$$x_{0j} = x_{i0} = 0 \qquad \text{für alle } i,j$$

$$x_{N+1\; j} = x_{i\; M+1} = 0 \qquad \text{für alle } i,j$$

$$x_{ij} \in \{0,1\} \qquad \text{für alle } (i,j) \in \mathcal{F} \setminus \mathcal{C}$$

- **Richard Kaye (2000) has proved that SAT (logical satisfiability problem) can be reduced in polynomial time to the Minesweeper Consistency Problem (MCP). Therefore, MCP is NP-complete.**

A geometric problem

- Take a sphere *S* of radius 1. Is it possible to add another 12 spheres with radius 1 in three dimensional space in such a way that all of them touch S?
  - this is indeed possible,
  - there is an infinite number of solutions.
- Is it also possible to solve the same problem with 13 instead of 12 spheres?
  - this is impossible!
- Model:

$$\|x_i\|_2^2 = 2 \qquad \text{für alle } i = 1, \ldots, N$$

$$\|x_i - x_j\|_2^2 \geq 2 \qquad \text{für alle } i \neq j \in \{1, \ldots, N\}$$

$$x_i \in \mathbb{R}^3 \qquad \text{für alle } i = 1, \ldots, N$$

- What is the smallest possible radius of S such that 13 spheres of radius 1 can touch?

Kepler's problem

- **The 18th Hilbert Problem (Part C):**
  Ich weise auf die ... Frage hin, wie man unendlich viele Körper von der gleichen vorgeschriebenen Gestalt, etwa Kugeln mit gegebenem Radius ..., im Raume am dichtesten einbetten, d.h. so lagern kann, daß das Verhältnis des erfüllten Raumes zum nichterfüllten Raum möglichst groß ausfällt.
- **Kepler conjectured, that the cubic face centered packing is the densest possible packing of spheres in three dimensions.**
- **1953 László Fejes-Tóth: finite dimensional optimization problems are sufficient.**
- **1990 Wu-Yi Hsiang: Proof, incomplete, in parts too unclear.**
- **1999 T. Hales: Computer-assisted proof, likely correct but very bulky and quite difficult to check, using global optimization techniques.**

Robotics

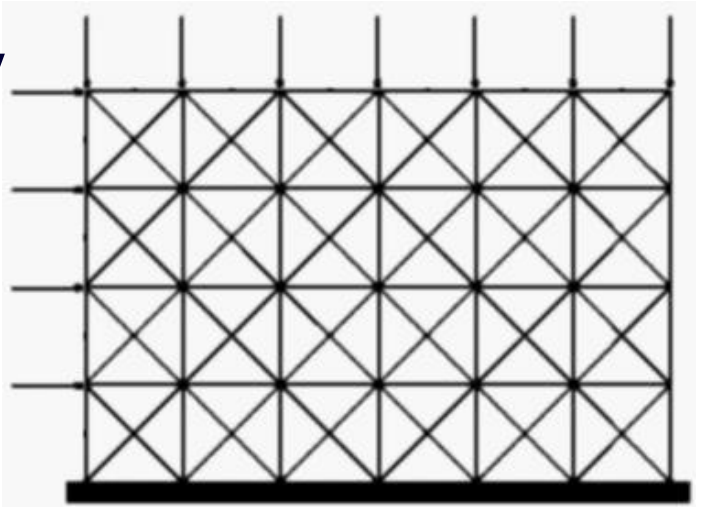
- Parallel robot at the European Synchrotron Radiation Facility in Grenoble
- Can handle up to 1000kg
- Accuracy better than 1μm

- Guarantee stability
- Compute the work space
- Compute the current position

- Design problem for robots

Worst-Case Analysis of structures with FEM

- Linear FEM equations become very difficult if the materials involved have uncertain material data (in reality 10-20% in elasticity and 1-5% in diameter).



- State-of-the-Art is Monte-Carlo-Simulation, which in general underestimates the worst case, sometimes by orders of magnitude.

- Industry relevant problems usually involve many thousand variables.

Global Optimization (GOP)

- **Mathematical Formulation:**

$$\begin{aligned} \min\ & f(x,z), \\ \text{s.t.}\ & F(x,z) = 0, \\ & x \in \mathbf{x}, \\ & z \in \mathbf{z}. \end{aligned} \qquad (x,z) \in \mathbb{R}^n \times \mathbb{Z}^m$$

Classes of solution algorithms

- **incomplete**
  Clever heuristic methods without safeguards, usually finding good feasible points quite fast (e.g. genetic algorithms),
- **asymptotically complete**
  Find the global optimum with probability 1 after running arbitrarily long (e.g. simulated annealing),
- **complete**
  Reaches the global optimum with guarantee, if no rounding errors happen and the running time is not limited. After finite time an approximation of the optimum within tolerances is found (branch and bound methods, e.g., BARON, LINGO, COCOS,...),
- **rigorous**
  Finds the global optimum with guarantee (except in degenerate cases) (e.g. GLOBSOL, Gloptlab)

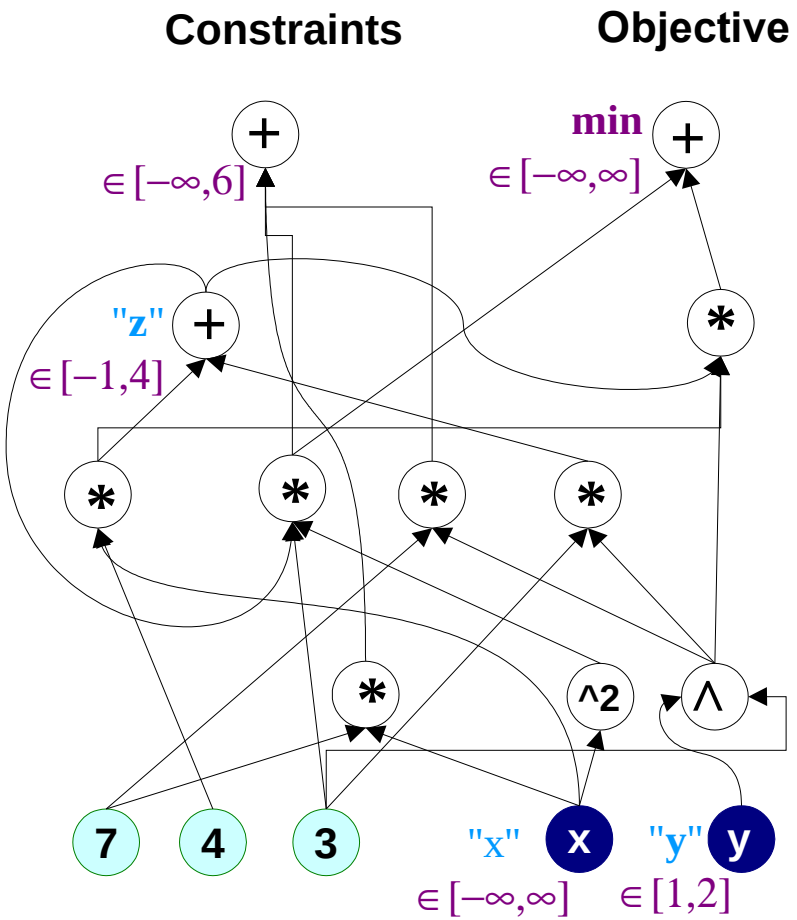
Internal Mathematical Representation

- The mathematical representation of a problem is

$$\begin{aligned} \min \quad & f(x) \\ s.t. \quad & F(x) \in S_c, \quad x \in S_v \end{aligned}$$

  where (currently) the sets S are boxes, $f = 0$ is allowed.
- The algorithmic representation is in graph form using not a tree (or forest) as usual but a directed acyclic graph (DAG).
- The DAG is simplified so that every subexpression is contained only once.
- The DAG is similar to a parse tree. Every node represents a mathematical operation (e.g. +,*,exp,...).

Directed Acyclic Graph (DAG)



- **DAG representation of the GOP**

$$\min\ 3x^2z+4xy^3z$$
$$\text{s.t. } z=4x+3y^3$$
$$\text{s.t. } 7x+3x^2z+7y^3\le 6$$
$$\text{s.t. } y\in[1,2],z\in[-1,4]$$

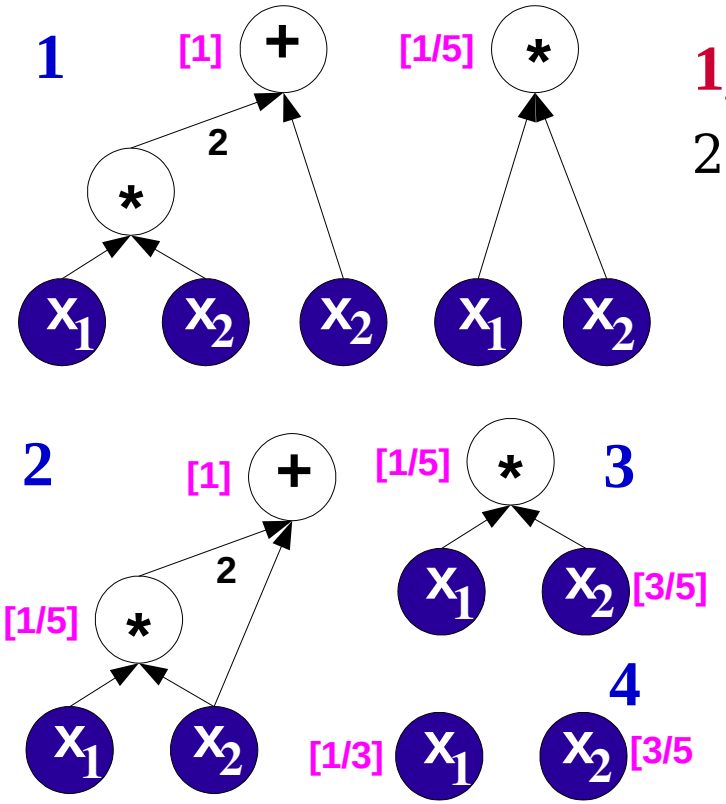
  - similar to a computational tree
  - a node may have more than one parent
  - constants and variables are sources (roots)
  - constraints and objective function are sinks (leaves)

Evaluation of a DAG

- Evaluation works similar to compute trees by performing a graph walk.
- Caching keeps evaluation work minimal.
- The whole problem is stored in one graph.
- Defining short-cuts makes it possible to replace graph walks by evaluation functions. Short-cuts may be defined at every node.
- Additional elementary functions can easily be incorporated.

COCONUT

DAG Simplification



1, 2

$$2x_1x_2+x_2=1$$
$$x_1x_2=1/5$$

3

$$x_2=3/5$$
$$x_1x_2=1/5$$

4

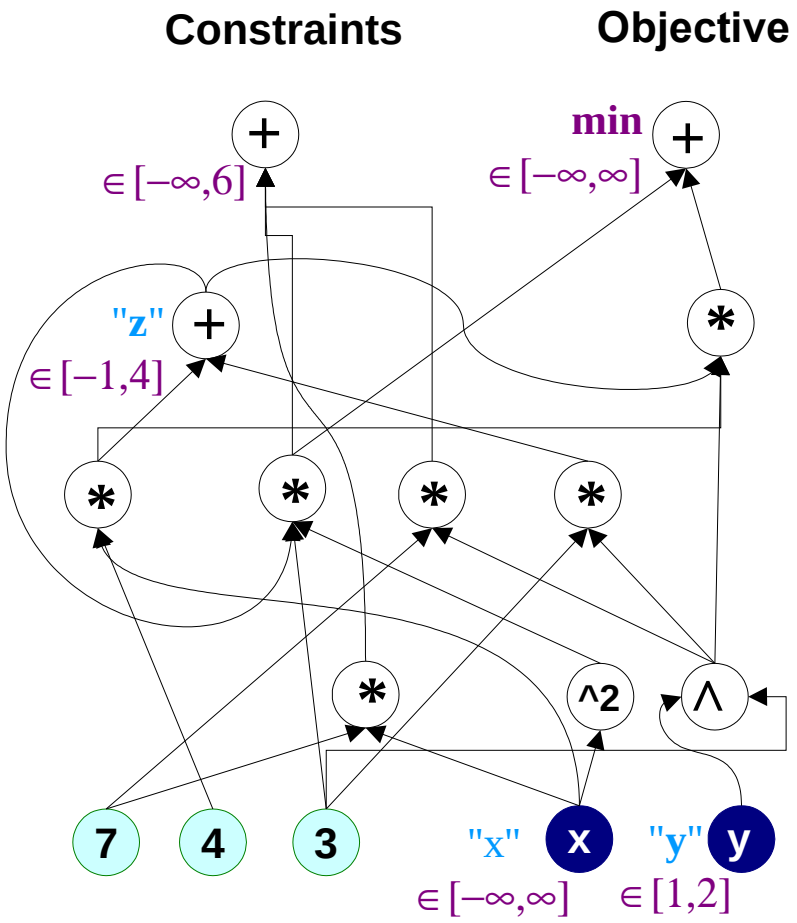
$$x_2=3/5$$
$$x_1=1/3$$

1. Common subexpressions are detected.
2. Constants and equality constrained subnodes are propagated.
3. Simple Sums are resolved.
4. Simple Products are resolved.

DAG - different interpretations 1

- Without change a DAG can be interpreted in different ways to suit the various algorithms.
  - the original most compact form has the smallest number of variables and it reduces the number of non-linear equalities $\Rightarrow$ local optimization
  - interpreting all named vertices as variables leads to a representation which is equivalent to the original problem definition.
  - to improve the sparsity pattern of the Hessian, additional vertices can be regarded as variables, hereby increasing sparsity as well as dimension.

DAG Interpretation (compact)

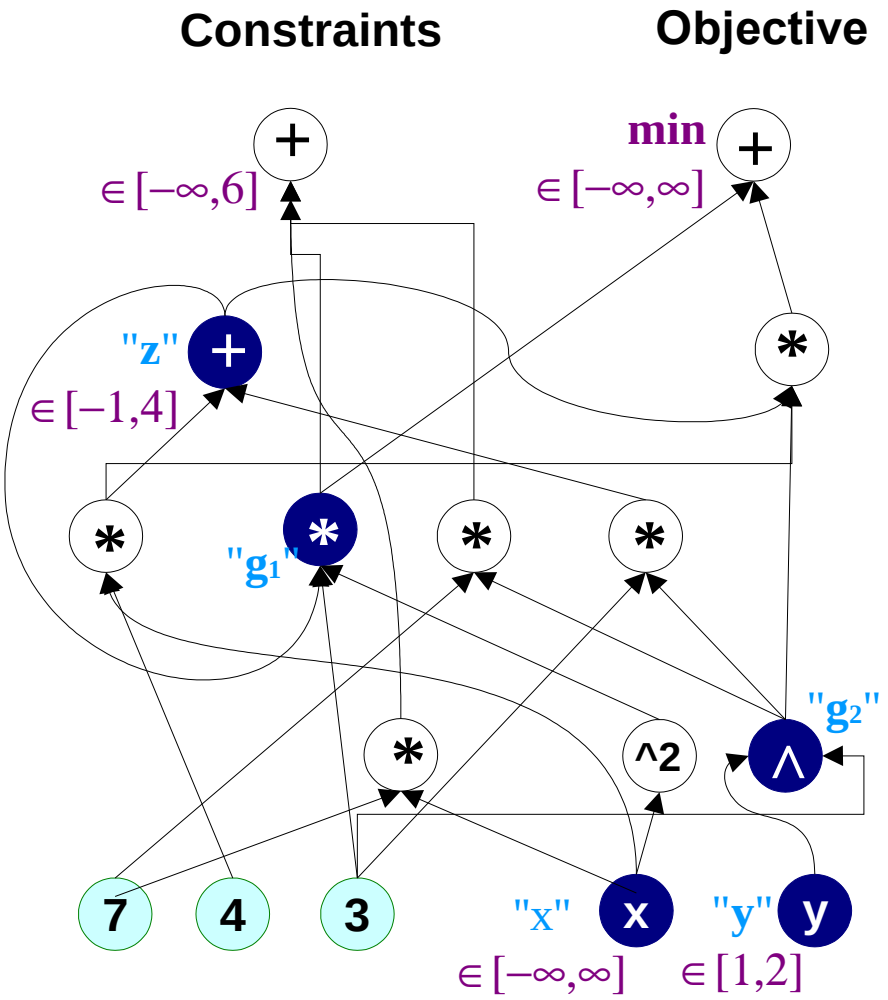


- **The true mathematical form of the problem represented by the example DAG is**

$$\min\ (3x^2+4xy^3)(4x+3y^3)$$
$$\text{s.t. } 7x+3x^2(4x+3y^3)+7y^3\le 6$$
$$\text{s.t. } -1\le 4x+3y^3\le 4$$
$$\text{s.t. } y\in[1,2]$$

- **It has lowest possible dimension 2 and no non-linear equalities.**

DAG Interpretation (sparse)

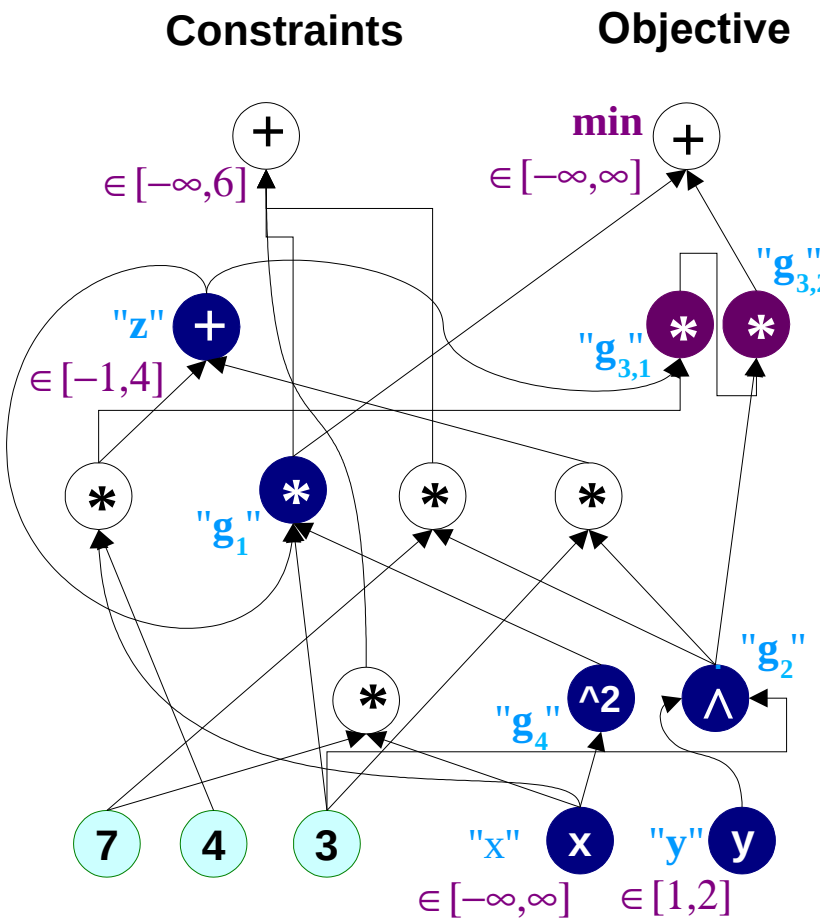


- **The DAG on the left corresponds to the problem**

$$\min\ g_1+4g_2xz$$
$$\text{s.t. } 4x+3g_2=z$$
$$\text{s.t. } 7x+g_1+7g_2\le 6$$
$$\text{s.t. } 3x^2z=g_1$$
$$\text{s.t. } y^3=g_2$$
$$\text{s.t. } y\in[1,2],z\in[-1,4]$$

where the g_i are variables "generated" by reinterpretation of vertices.

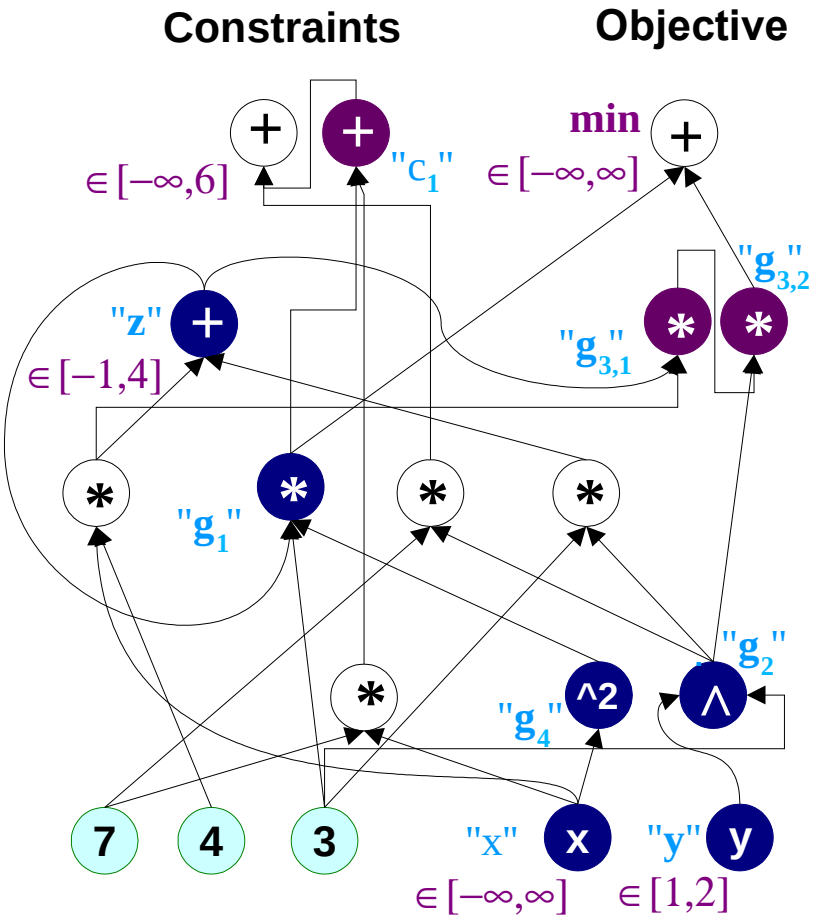
DAG Interpretation (quad+sep.)



- **Generating quadratic+ separable form requires implicit vertex splitting (e.g. in Gloptlab).**

$$\min\ g_1+g_{3,2}$$
$$\text{s.t. } 4x+3g_2=z$$
$$\text{s.t. } 7x+g_1+7g_2\le 6$$
$$\text{s.t. } 3g_4z=g_1$$
$$\text{s.t. } y^3=g_2$$
$$\text{s.t. } 4xz=g_{3,1}$$
$$\text{s.t. } g_{3,1}g_2=g_{3,2}$$
$$\text{s.t. } x^2=g_4$$
$$\text{s.t. } y\in[1,2],z\in[-1,4]$$

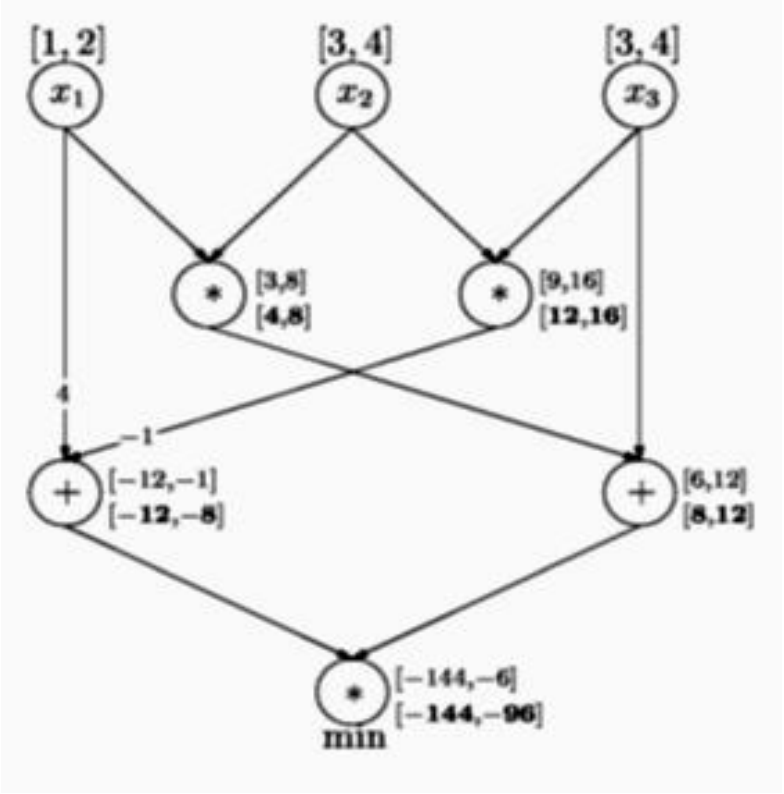
DAG Interpretation (ternary)



- **For a ternary representation the sums have to be implicitly split, too.**

$$\min\ g_1+g_{3,2}$$
$$\text{s.t. } 4x+3g_2=z$$
$$\text{s.t. } c_1+7g_2\le 6$$
$$\text{s.t. } 7x+g_1=c_1$$
$$\text{s.t. } 3g_4z=g_1$$
$$\text{s.t. } y^3=g_2$$
$$\text{s.t. } 4xz=g_{3,1}$$
$$\text{s.t. } g_{3,1}g_2=g_{3,2}$$
$$\text{s.t. } x^2=g_4$$
$$\text{s.t. } y\in[1,2],z\in[-1,4]$$

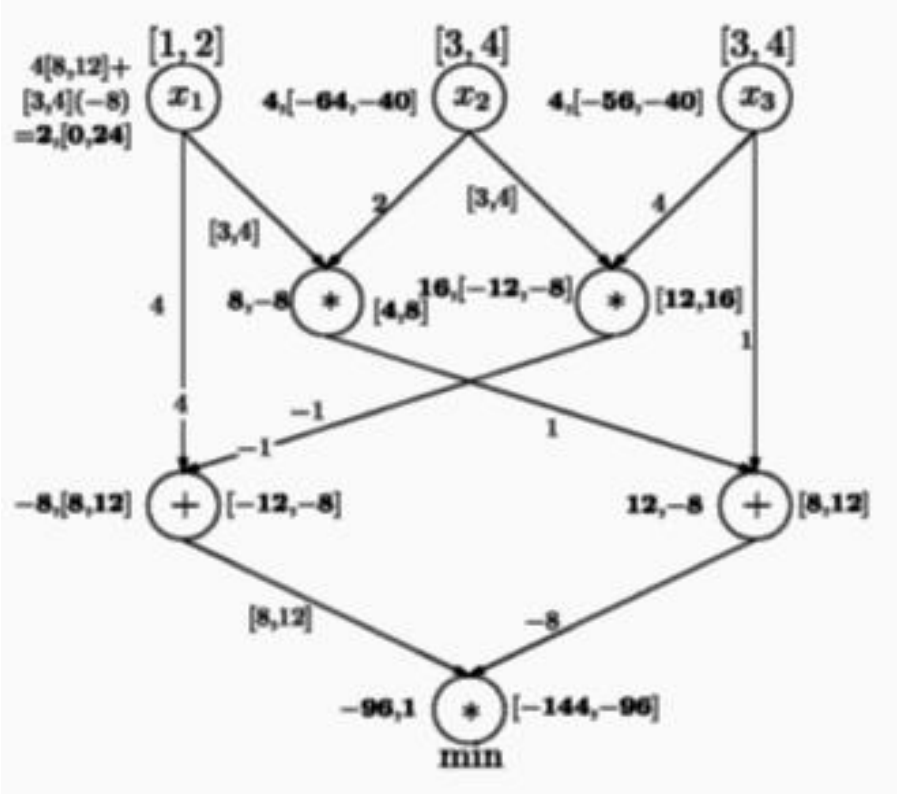
Example



- **Interval evaluation and constraint propagation produce bounds on each node**
- **No reduction on the domain of the variables**
- **The bounds on intermediate nodes are improved compared to interval evaluation**

$$\min\left(x_1 x_2 + x_3\right)\left(4x_1 - x_2 x_3\right),\quad x_1 \in [1,2],\quad x_2, x_3 \in [3,4]$$

Example (ctd.)



- **Linear enclosures produced using slopes give redundant constraints, e.g.**

$$24(x_1-2)-48(x_2-4)-32(x_3-4)\leq 0$$

Constraint Propagation

- Known bounds on the graph's nodes are combined by using well-known estimates for the operations and their partial inverses.
- For $h=f\left(g_1,g_2,\ldots,g_k\right)$ we define the forward propagation

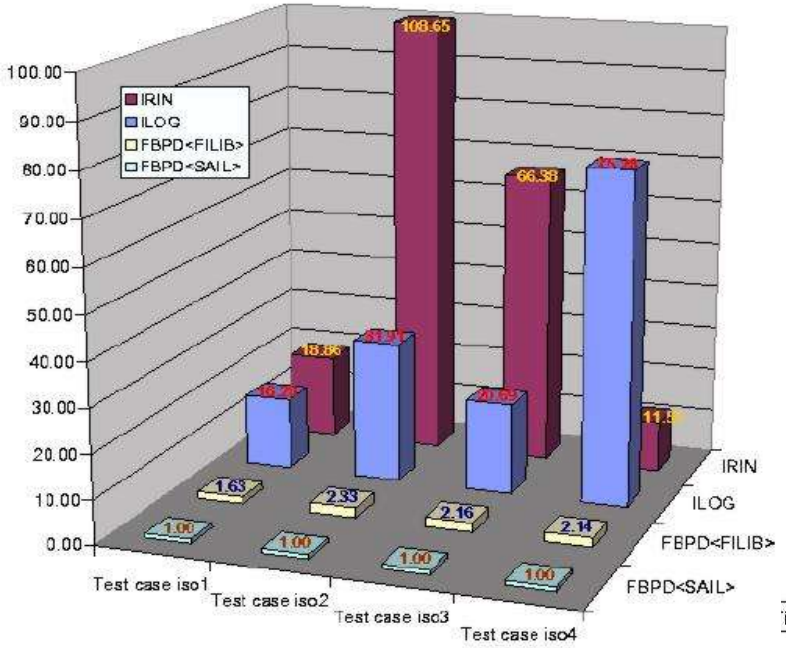
$$h^{(n+1)}=f\left(g_1^{(n+1)},g_2^{(n+1)},\ldots,g_k^{(n+1)}\right)\cap h^{(n)}$$

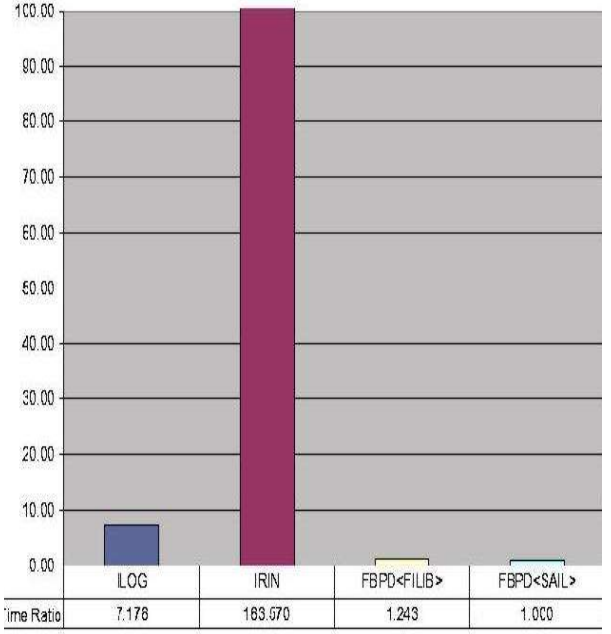
and the backward propagation

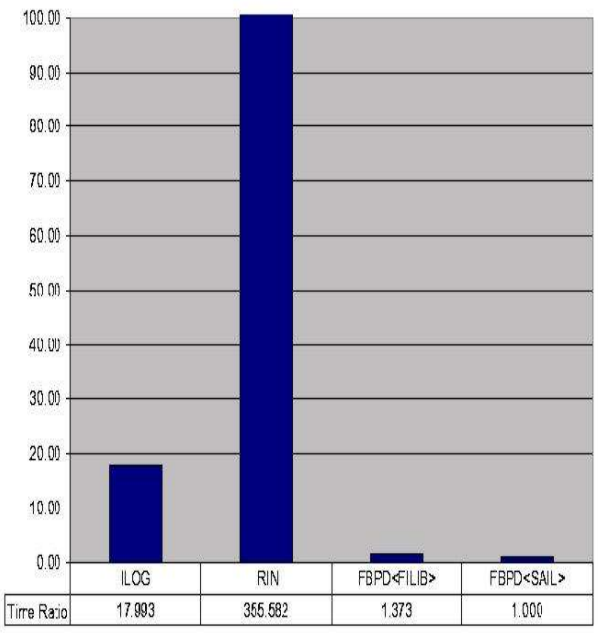
$$g_j^{(n+1)}=f^{-1,j}\left(g_1^{(n+1)},g_2^{(n+1)},\ldots,g_{j-1}^{(n+1)},g_{j+1}^{(n)},\ldots,g_k^{(n)}\right)\left(h^{(n+1)}\right)\cap g_j^{(n)} \quad .$$

- Iteration of the two propagation methods improves the estimates for all nodes.

Constraint Propagation

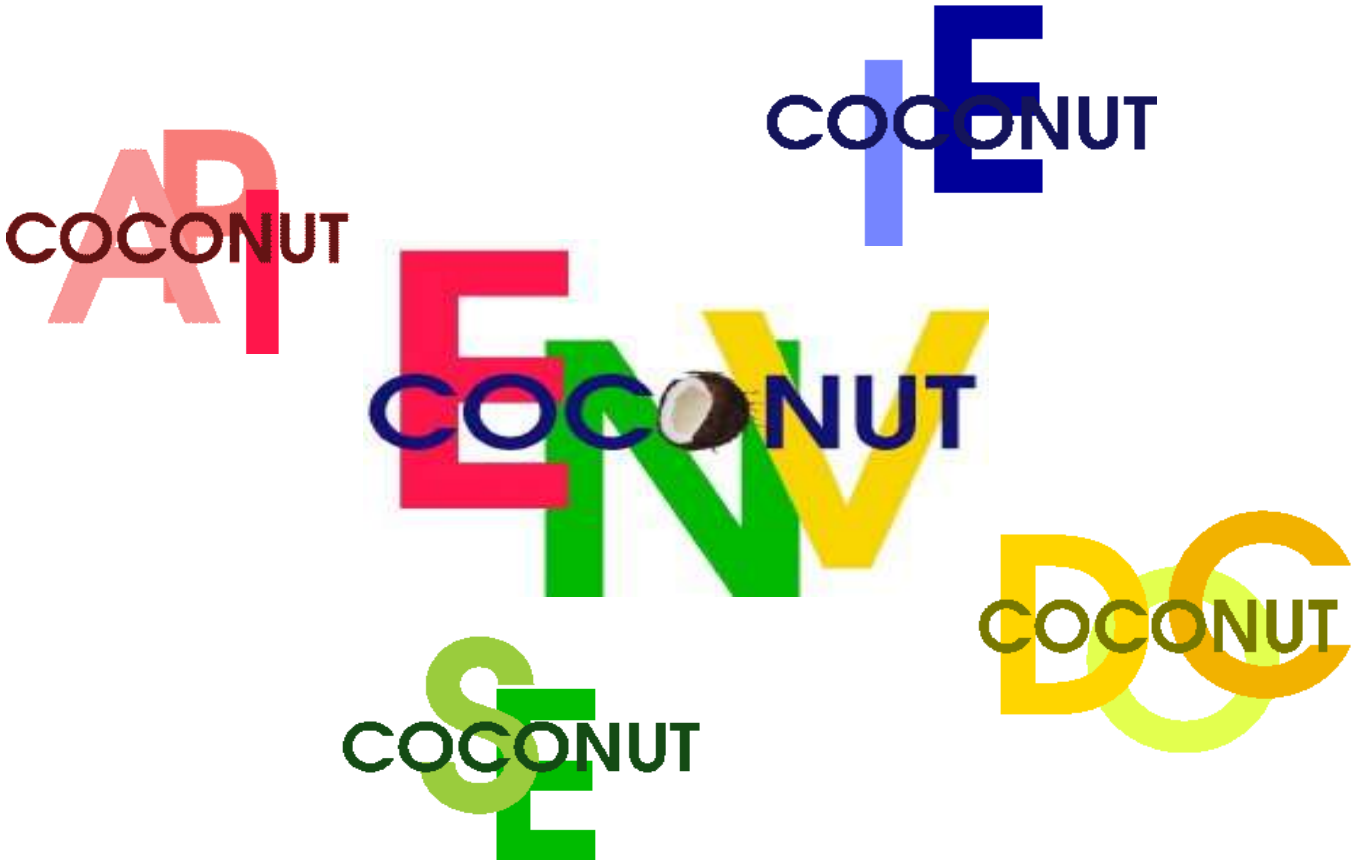






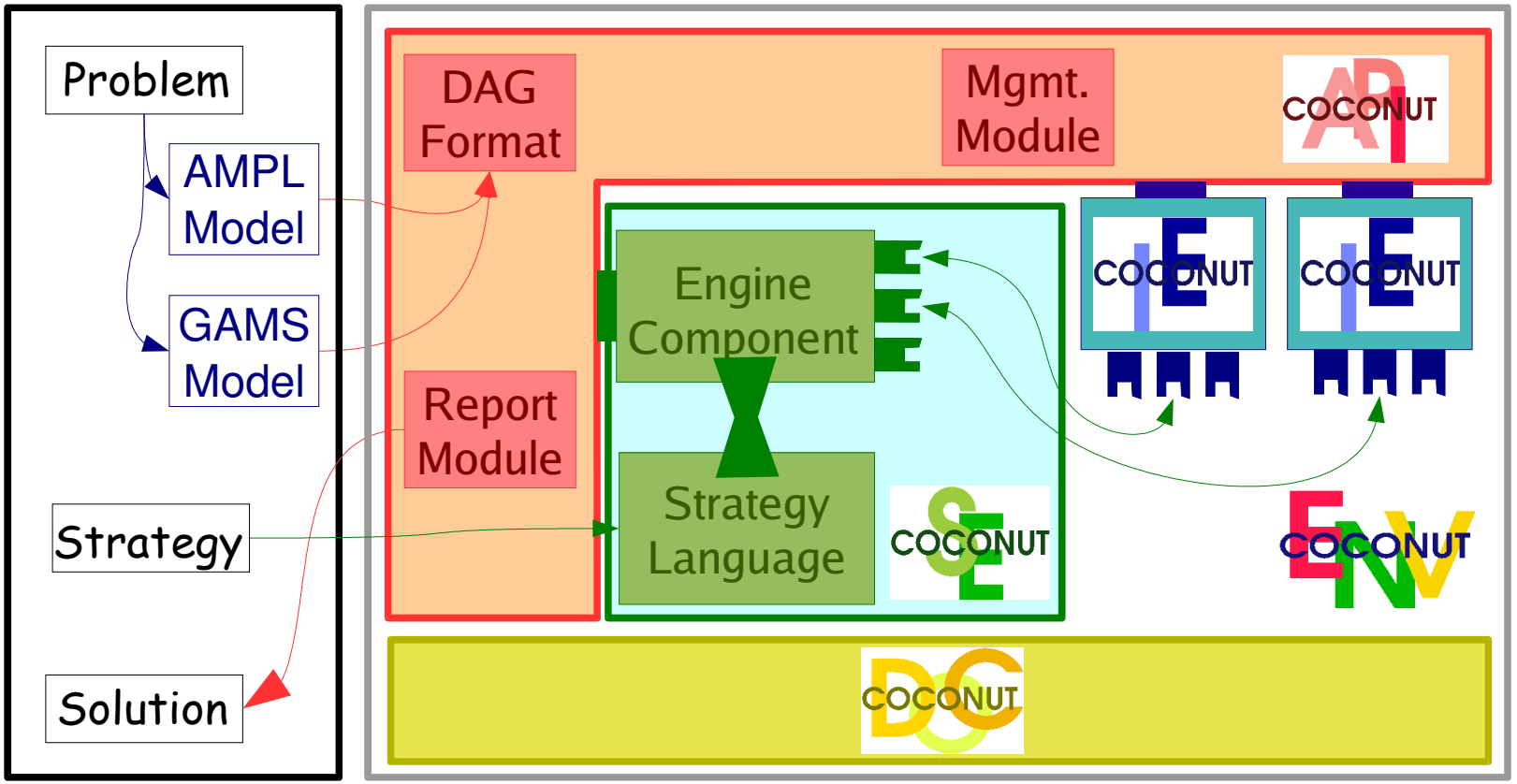
- In these tests we compare the DAG based constraint propagator (with Xuan-Ha Vu, EPFL) with other state-of-the-art algorithms.
- Hull is one of the best public domain programs (IRIN).
- ILOG Solver is the best known commercial solver.
- FBPD ist 7-75(18) times faster than (tuned!) ILOG Solver and 5-600(150) times faster than IRIN Hull.

The COCONUT environment




COCONUT

Design


Problem
AMPL Model
GAMS Model
Strategy
Solution
DAG Format
Report Module
Mgmt. Module
API
COCONUT
Engine Component
Strategy Language
SE
COCONUT
COCONUT
COCONUT
COCONUT
DC
COCONUT

Basic Modular Setup

Strategy Eng.

main part of algorithm,
searches, makes decisions,

Inference Mod.

structure analysis, bisection, local optimization, interval analysis, constr. propagation, linear relaxation, convex opt., search graph analysis,...

Report Mod.

file generation (AMPL, GAMS, C, Fortran90,...), human readable output, checkpointing

Management M.

problem management, search graph management, resource management

IST-2000-26063 COCONUT

Project funded by the Future and Emerging Technologies arm of the IST Programme
FET-Open scheme

COCONUT API

Modular design

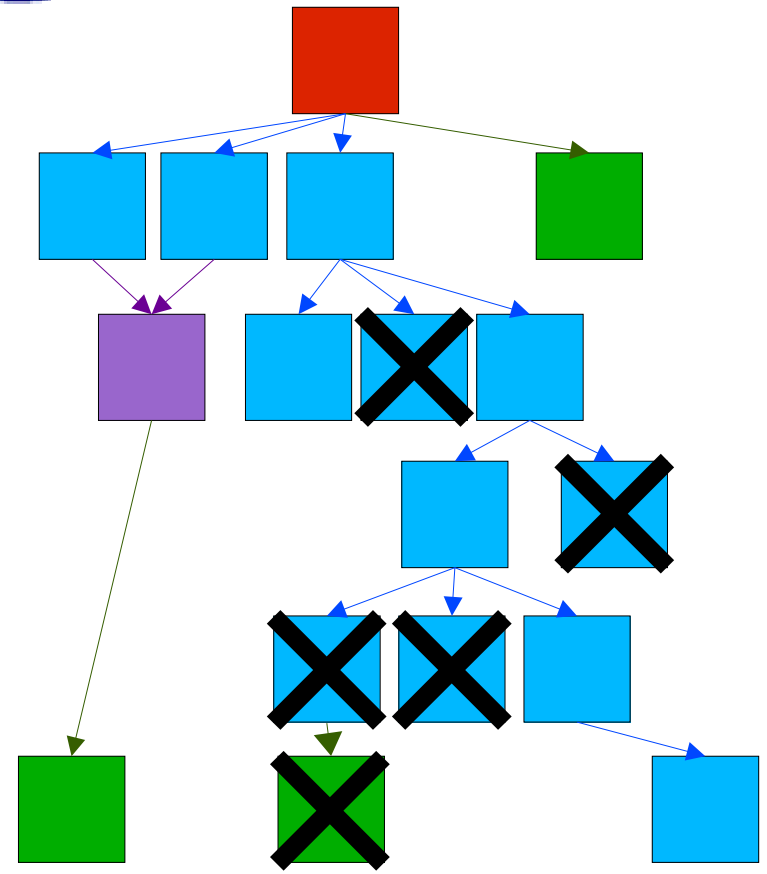
- The API is designed to make the development of the various module types independent of each other and independent of the internal model representation.
- Provides data abstraction, basic strategy structures.
- A collection of C++ classes.
- Supports dynamic linking.

COCONUT API

Modular design (ctd.)

- Five different module types, all modules are subclasses of one of these classes.
- Communication with the strategy engine by the search database and standard control structures.
- The API implementation (w/o modules) consists of 80000 lines of C++ code and a few perl scripts, organized into 250 files, 5 MB of disk space.
- Available modules: 70000 lines (2 MB) without libraries, Strategy engine: 49800 lines (1.73 MB).

Search graph



- Starts at the original model
- Contains relaxations
- and splits.
- It is not a tree since it might also contain glueings.
- Some of the nodes will be terminal, since they can be solved completely.

COCONUT

Model Relations

$$\begin{aligned} \min \quad & c^\top x \\ \text{s.t.} \quad & b^\top x \le b_0 \\ & a(x) \le 0 \\ & x \in [x] \end{aligned}$$

Relaxation

$$\begin{aligned} \min \quad & c^\top x \\ \text{s.t.} \quad & b^\top x \le b_0 \\ & x \in [x] \end{aligned}$$

Reduction
e.g. Add cut, prune box

$$\begin{aligned} \min \quad & c^\top x \\ \text{s.t.} \quad & b^\top x \le b_0 \\ & d^\top x \le d_0 \\ & a(x) \le 0 \\ & x \in [x] \end{aligned}$$

Split

$$[x'] \cup [x''] = [x]$$

$$\begin{aligned} \min \quad & c^\top x \\ \text{s.t.} \quad & b^\top x \le b_0 \\ & a(x) \le 0 \\ & x \in [x'] \subset [x] \end{aligned}$$

$$\begin{aligned} \min \quad & c^\top x \\ \text{s.t.} \quad & b^\top x \le b_0 \\ & a(x) \le 0 \\ & x \in [x''] \subset [x] \end{aligned}$$

Internal Representation

- Models (subproblems) are organized in the search graph, represented by a Directed Acyclic Graph (DAG).
- Most nodes in the search graph contain deltas, i.e. *differences* between models.
- Additional information for models can be stored in the search database in a flexible way.

Search graph

- The search graph has one focus onto the search node for each model, which is analyzed by the algorithm.
- Every model in the search focus is copied into an enhanced structure – the work node. A reference to this work node is passed to the inference engines.
- The graph itself can be analyzed by graph analyzers, modules which use search inspectors.

COCONUT API

Search database

- The search database stores the deltas, the model annotations, and whatever data the inference engines want to store.
- It is organized like SQL databases in tables with columns and rows.
- The columns can hold (almost) arbitrary C++ objects, even methods which compute the column values dynamically depending on the evaluation context.
- For every worknode a view of the database is created, restricting the database to a subset and a specific evaluation context.

Inference Modules

- Two types of inference modules are designed:
  - inference engines analyze work nodes:
    - Model analysis (e.g. find convex part)
    - Model reduction (e.g. pruning, fathoming)
    - Model relaxation (e.g. linear relaxation)
    - Model splitting (e.g. bisection)
    - Model glueing (e.g. undo excessive splitting)
    - Update local information (e.g. local optimization)
  - graph analyzers analyze the search graph:
    - Box selection
    - Gluing
- Inference modules never change the model but calculate which changes may be made and are considered useful.

Inference Modules: General features

- All inference modules only advertise changes.
- There is a fixed documentation structure defined.
  - Services Provided
  - Limits
  - Structure, Prerequisites of Input
  - Structure, Features of Output
  - Control Parameters
  - Termination Reason
- They produce a result where every possible change is listed together with a weight (the higher the weight the more important the change).
- They collect statistical data to support the strategy engine in making decisions.

Inference Engines implemented as State of the Art

- Several state of the art techniqnes were implemented as inference engines:
  - Starting point generator (STOP, PGSL, Simple)
  - Local optimization (DONLP2-INTV, IPfilter)
  - Karush-John-Condition Generator
  - Point Verifier
  - Simple convexity detection
  - Exclusion Box
  - Interval constraint propagation (2 versions)
  - Linear Relaxation
  - LP solver (CPLEX, XPRESS-MP, LP_solve)
  - Basic Splitter
  - BCS (box covering solver)

Contributions from the outside of the COCONUT project

We are happy that researchers and companies from outside the COCONUT project agreed to complement our efforts in integrating the known techniques:

- Bernstein modules by J. Garloff & A. Smith (U. Konstanz)
- Verified lower bounds for convex relaxations by Ch. Jansson (TU Hamburg-Harburg)
- GAMS reader by the GAMS consortium
- Taylor arithmetic by G. Corliss (Marquette U.)
- Asymptotic arithmetic by K. Petras (U. Braunschweig)
- XPRESS-MP commercial LP-solver (Dash Optimization)
- GLOBAL – Heuristic global optimizer (T. Czendes, U. Szeged)
- PGSL – Heuristic global optimizer (B. Raphael, EPFL)

Report Modules

- This class of modules produces output. Various types of files and human readable output will have to be created.
- Examples:
  - Solution Report (for humans, AMPL, GAMS,...)
  - Conversion to other solvers or input formats (BARON, LGO, C, Fortran90, GlobSol, LINGO, Frontline Interval Solver, Gloptlab)
  - Structure Report
  - Progress Information
  - Checkpointing
  - Debugging output

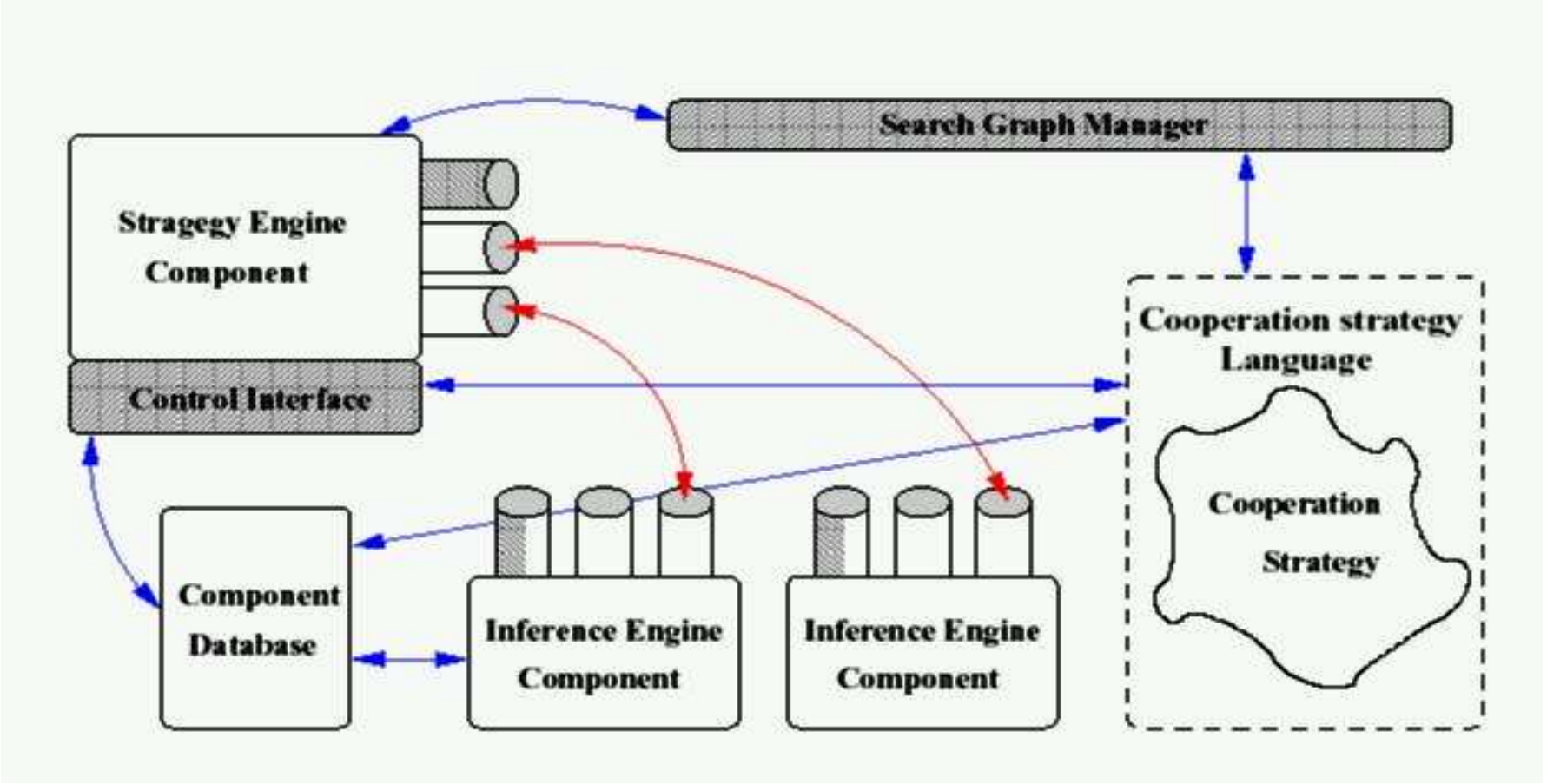
Management Modules

- Corresponding to every internal part of the program, a class of management modules is designed. All management modules are derived from two base classes.
  - Management modules
    - Model management
    - Data collection
    - Resource management
  - Initializers
    - Initialization management
    - Destruction management
- Management modules never calculate anything. They just perform some of the changes which have been advertised by inference modules.

COCOS
combination strategy

- There are two versions of the solution strategy COCOS
  - Hardcoded, which has to be recompiled and linked
  - Strategy Engine based, which can be changed dynamically
- Presolve phase at the root node
- Main phase (Branch & Bound)
- Postsolve phase
  - Clean up and postanalysis of all small boxes found

Strategy Engine



Strategy Engine

- It is the core of the algorithm
- It calls the management modules, the report modules, and the inference modules in succession.
- It can be programmed using a strategy language (interpreted, Python based).
  - (Semi-)interactive and automatic solution process
  - Debugging and single-stepping of strategies
  - Object oriented, dynamically typed objects, garbage collected
  - Easily extendable

Strategy Engine (ctd.)

- Manages the search graph via the search graph manager,
- Manages the search database via the database manager,
- Uses a component framework (CORBA based) to communicate with the inference engines,
- Launches inference engines dynamically (on need) to avoid memory overload,
- Provides a management interface,
- Strategy engine is itself a component, so multilevel strategies are possible,
- Prepared for distributed and parallel computing, and distributed memory

Extensibility

- The strategy language makes it easy to change the strategy.
- A registration phase during initialization removes the need to recompile the program when new inference engines are added.
- Registration also allows us to balance scientific and commercial interests:
  - Free but reduced core version with open API specification
  - Free strategy engine with basic strategy
  - Advanced commercial components
- Extending the system by external contributers is made easy by this modular design.

Open Problems

- Use global optimization methods for solving the following problems
  - Protein folding,
  - minimizing the Gibbs free energy function (Chemistry),
  - Robotics design.
- Adding ODE and PDE constraints to global optimization algorithms (necessary, e.g., for trajectory design, aircraft design,...).
- Adding black-box-functions to branch-and-bound global optimization algorithms.
- Automatically removing symmetries from GOPs.
- Exploiting sparsity efficiently.
- etc.

Questions and Discussion

Thank you!