Towards the Coevolution of Incentives in BitTorrent

Tamás Vinkó and David Hales

University of Szeged, Department of Computational Optimization
Árpád tér 2, H-6720 Szeged, Hungary
tvinko@inf.u-szeged.hu, daphal@inf.u-szeged.hu

Abstract: BitTorrent is a peer-to-peer file sharing system that is open to variant behavior at the peer level through modification of the client software. A number of different variants have been released and proposed. Some are successful and become widely used whereas others remain in a small minority or are not used at all. In previous work we explored the performance of a large set of client variants over a number of dimensions by applying Axelrod's round-robin pairwise tournament approach. However, this approach does not capture the dynamics of client change over time within pairwise tournaments. In this work we extend the tournament approach to include a limited evolutionary step, within the pairwise tournaments, in which peers copy their opponents strategy (client variant) if it outperforms their own and also spontaneously change to the opponents strategy with a low mutation probability. We apply a number of different evolutionary algorithms and compare them with the previous non-evolutionary tournament results. We find that in most cases cooperative (sharing) strategies outperformed free riding strategies. These results are comparable to those previously obtained using the round-robin approach without evolution. We selected this limited form of evolution as a step towards understanding the full coevolutionary dynamics that would result from evolution between a large space of client variants in a shared population rather than just pairs of variants. We conclude with a discussion on how such future work might proceed.

Keywords: peer-to-peer; BitTorrent; selection rules; design space analysis

1 Introduction

Many popular content sharing systems are based on peer-to-peer (P2P) technology. In P2P systems the participating users, also called peers, are the content providers and demanders at the same time. Among these systems, BitTorrent [1] is the best representative example, used by millions of Internet users every day. In a BitTorrent system, peers can use different clients, i.e. a

computer program which runs the BitTorrent protocol, or a modified version of it. Modification usually means some extension upon the original protocol in order to provide the user with better quality of service. Apart from the protocol modifications actually deployed as clients –for example uTorrent, VuZe, Tribler, etc.–, many interesting variants were proposed in the scientific literature of BitTorrent systems, see, e.g. [2, 3, 4, 5]. Most noticeably, the paper of Rahman *et al.* [6] gives a list of more than 3000 protocol variants. However, the aim of the paper was to lay down the methodology of Design Space Analysis (DSA) of distributed protocols for measuring their performance, robustness and aggressiveness.

In this paper we give an extension of the DSA concept by studying the effects of applying evolutionary approaches to the peers.

1.1 Motivation

BitTorrent and open peer-to-peer protocols in general require the cooperative interaction of individual peers if they are to function optimally. This is because performance is collectively produced yet actions are individually selected. Since each individual peer may run a different client variant (so long as it implements the conventions of the specified protocol) there is the possibility for strategic interaction and consequent collective action problems. In the context of BitTorrent this involves free riding (downloading data, but not uploading data) rather than sharing data.

However, this problem is quite general in any open distributed system where the client software that runs on each peer is not under the control of a central authority or designer.

Consequently, designers of open peer-to-peer protocols (such as BitTorrent) include incentive mechanisms to encourage cooperation. In its simplest form this involves client software which punishes other peers who do not operate in a cooperative way. If a sufficient number of peers execute such a client, then free riding can be controlled since it is not then in the individual interests of a single peer to change to free riding behavior.

This general problem of cooperative collective action has been addressed through game theory and computer simulations. Axelrod [7] used the pairwise Iterated Prisoner's Dilemma game and computer programs implementing game strategies to perform a "round-robin tournament" (RRT) to examine which individual strategies did well on average against all other strategies.

For a given P2P application it is possible to view client variants as strategies in a complex multiplayer game. In general, such games are too complex to be tractable within analytical game theoretic frameworks without gross simplifications and

assumptions that do not hold in the real world. Hence, computer simulation is often used.

In previous work we applied a form of Axelrod's RRT (which we called DSA) to the BitTorrent protocol identifying a number of interesting, counterintuitive and high performance variants. This approach involved taking every pair of possible client variants (or strategies) and performing simulation runs in which the population was partitioned into two fixed size subpopulations running the two different variants. Both variants then mixed freely during interactions which involved downloading and uploading file pieces following the BitTorrent wire protocol. Results for each strategy were calculated based on average performance of that strategy against all other strategies by aggregating all of the relevant simulation runs and calculating several statistics.

We modify the RRT such that rather than only examining the performance of one strategy (or client variant) against another in *fixed* subpopulations we allow clients to apply evolution such that they can copy and mutate variants from others who outperform them. Hence, this approach allows for relative sizes of the two subpopulations to change during interaction over time within a tournament.

This extension is motivated by several issues: firstly, are the original DSA results reproduced when peers are given the ability to evolve directly? Secondly, do those strategies previously identified as robust against other strategies evidence evolutionary robustness; and thirdly, how do different evolutionary algorithms effect the outcomes?

Hence, in this work we are extending the previous DSA approach by applying evolution *within* each tournament between only pairs of strategies. We are not applying general evolution to the entire space of strategies in one large population.

This allows us to test if the original DSA results are consistent with evolution applied to pairs of strategies and to examine in more detail the robustness and performance of client variants. We consider this *limited* evolutionary approach as a step towards a full coevolutionary analysis over a large strategy space while maintaining the ability to produce meaningful insights.

This approach applies a limited form of evolution between only pairs of solutions (client variants) because it extends the round-robin tournament approach. Hence, it is not possible to apply operators such as crossover since these would introduce more than two variants into the tournament. Also, for the same reason mutation is implemented as swapping to the other client variant rather than producing a new client variant. It would be of interest to compare the results from a full co-evolutionary analysis in which all variants compete and evolve within a single population. However, based on previous results, as discussed in section 1.2, we expect such an approach to produce highly variable and difficult to analyse results. This is because coevolutionary dynamics over complex strategies tend to lead to cyclical and highly contingent dynamics such that, in the worst case, outcomes

may appear indistinguishable from noise. However, we discuss ways forward in this regard in the conclusion.

1.2 Related Work

Several previous works have studied the application of incentive mechanisms to regulate open distributed systems. This area of research has been termed distributed computational mechanism design [8]. Approaches essentially fall into two broad categories. Firstly, those that apply analytic game theoretic formulations [9, 10, 11] and secondly, those that utilize simulation approaches [12, 13, 14, 15]. In general, analytic approaches require high levels of abstraction that limit their applicability to the design of realistic deployable protocols. Alternatively, simulation can be applied to highly realistic scenarios, but often lacks detailed sensitively analysis due to the large parameter spaces of real protocols. Previous work attempted to balance these two aspects via the use of simulation over a large yet tractable space of parameters related to realistic protocol designs [6]. We build on this latter work in the present article. Jin *et al.* [16] presented an evolutionary analysis of BitTorrent P2P protocol variants by simulating the coevolution of six existing deployed variants within a single population. Our work differs in that we investigate on the coevolution between two variants at a time in the population. However, by limiting evolution in this way, to within tournament pairs, we are able to examine all coevolutionary pairings between over 500 individual protocols. Hence, through this abstraction we aim to move towards an understanding of the coevolutionary dynamics of more extensive forms of evolution. It is notable in the work of Jin *et al.* that the coevolutionary dynamics of populations composed of six protocol variants are difficult to analyze due to what appear to be highly contingent outcomes.

2 Background terminology

2.1 Design Space Analysis

Inspired by the seminal work of Axelrod [7], a method called Design Space Analysis (DSA) was proposed by Rahman *et al.* [6] to comprehensively model incentives in distributed protocols. This method models interactions between participating users playing repeated games. It combines the specification of a large design space of protocol variants together with their analysis done by simulations. The specification has two steps: (1) parametrization, which is the determination of the design space's dimensions, and (2) actualization, in which the actual values of the individual dimensions get specified. After these two steps, a solution concept can be used, where every element of the design space gets characterized by

different measures. DSA proposed three measures: Performance, Robustness and Aggressiveness – also called PRA quantification. Given a utility function, the Performance of a protocol is the average performance of the whole system under the assumption that all peers use the same protocol variant. The utility function is always domain specific. In a content distribution system the utility function is usually the average download speed of the users, but other measures could be used. The Robustness and Aggressiveness measures are defined in a system composed by different protocol variants and they indicate the ability of outperforming other protocol variants. By these three measures, the properties of all the protocols can be characterized as a three dimensional point.

Robustness indicates the ability of a given protocol variant to outperform its opponent variants (averaged over all tournaments). Hence, it measures how "robust" a given variant is to being dominated (i.e., outperformed) by other variants in a head-to-head tournament. In this sense it is a measure of relative performance a detailed description of how Robustness is calculated can be found in Section 4.3. We do not use the Aggressiveness measure in the present paper, but its definition can be found in [6].

2.2 BitTorrent

Maybe the most important idea behind the BitTorrent P2P protocol is that the files to be shared are divided into *pieces*. During the download of a particular file the peers (or nodes) obtain the pieces from a (usually dynamically changing) set of different nodes, which can consist of two types of users, leechers and seeders. *Leechers* are peers who are currently obtaining the file, i.e. those who do not have a copy of the entire file. *Seeders* are uploading exclusively, they do have all the pieces of the file. What makes the BitTorrent protocol highly scalable and very efficient is that the leechers can be uploaders as well. By default leechers follow a *rarest-first* rule to obtain pieces. Due to this, leechers have good chance to have pieces which can be traded for other pieces with other leechers, also called its neighbors. Each peer has upload capacity, which is divided into slots of two types: *regular unchoke* and *optimistic unchoke slots*. Regular unchoke slots are assigned to the subset R of neighbors which recently provided data with the highest speed. The assignments get re-evaluated in every unchoke interval, which is usually fixed to 10 seconds. On the other hand, optimistic unchoke slots are assigned to strangers (i.e., randomly selected neighbors), which also get re-evaluated in fixed period of time. In case an optimistically unchoked peer p is found to be faster than any of the regularly unchoked peers, then peer p is moved to the set R, replacing the slowest peer in R.

The set of leechers and seeders who exchange pieces of a particular file is called a *swarm*. A collection of swarms are called a community. Communities usually emerge around a central web server, called a *tracker,* which is an important part of the network. A peer upon joining a swarm it requests the IP addresses of other

peers participating in this swarm. The tracker can have other features like providing a searchable database of available content.

2.3 Parametrization

The protocol design space of BitTorrent can be spanned over the following dimensions:

Peer discovery: in order to participate and possibly interact with each other in the same swarm, peers need to find each other.

Stranger policy: this policy is applied when a peer is interacting with a previously unknown peer.

Selection function: this function decides which of the known peers should be selected for interaction.

Resource allocation: defines the way a peer divides its (upload) resources among the selected peers (which are given by the Selection function).

Using this design space a user of the BitTorrent network can enter with a client using either default protocol parameters or modified ones. Modification of the protocol can be motivated by aiming at improving the individual performance or even to trick the system by freeriding (i.e. being only a leecher and not uploading to any other leechers). We term a specific actualization of the parameters listed above a *strategy*.

3 Evolutionary approaches

In this section we give details of the evolutionary algorithm which can be used to find out the evolutionary behavior of a pair of unique protocols. The algorithm starts with N peers, and two strategies, A and B. Initially, half of the peers use strategy A and the other half use strategy B. The *fitness* of peer i is defined as $f_i = 1 - w + wU_i$, where U_i is the utility of peer i and $w \in [0,1]$ measures the intensity of selection (strong selection means $w = 1$ and weak selection means $w \ll 1$). By default, we measure the utility U_i as the average download speed in the current time interval, i.e., in the last R rounds.

The steps of the algorithm are the following:

Step 1 Let $f_i = 0$ for all $i = 1, \ldots, N$.

Step 2 For R rounds let the peers play the 'BitTorrent game'. Within a round, peers connect to each other (using a prescribed rule) and exchange pieces of the file they want to download.

Step 3 In each *R*th round apply a selection rule on *K* pairs of peers in order to update the composition of these peers' strategies.

Step 4 If the total number of steps equals to R_T then stop, otherwise go back to Step 1.

Note that *K* represents the *selective pressure* meaning how much selection is performed per round. A collection of possible *selection rules*, which can be applied in Step 3, will be discussed in Section 3.1. It is worth to note that with the choice $R=R_T$, $K=N$ and $w=1$we get back the original robustness test used in [6].

3.1 Selection rules

Now we give a list of selection rules that will be used later in the experimental part (Section 4). All of them use the parameters *K* and *m*, which must be set up at the beginning of the experiment. Note that in all rules we apply a *mutation* operator, which is as follows. With probability *m*, peer *j* switches its strategy to the opposite one. Technically, this means that we generate a random number *r* between 0 and 1 and if $r \leq m$ holds, then we switch.

Tournament selection

This involves repeated tournaments between pairs of randomly selected peers as applied in [17].

Repeat *K* times:

Select two peers, peer *i* and *j* uniformly at random, with replacement.

If $f_i \geq f_j$ holds, then

Set the strategy of peer *j* to be the same as the strategy of peer *i.*

Apply mutation on peer *j.*

Reset $f_j = 0$.

end if.

Death-birth updating

This involves selecting a peer randomly and setting its strategy based on the proportion of the strategies used in the entire population [18].

Repeat *K* times:

Select a peer *j* uniformly at random.

Change its strategy according to: $F_{A\vee B}/\left(F_A+F_B\right)$, where $F_A=\sum_{q\ uses\ A} f_q$, $F_B=\sum_{r\ uses\ B} f_r$, and $F_{A\vee B}$ equals to F_A if peer j uses strategy B, and equals to F_B if peer j uses strategy A.

Apply mutation on peer j.

Reset $f_j=0$.

Birth-death updating

This involves selection of an individual proportional to its fitness value and then its offspring is replacing another randomly chosen peer.

Repeat K times:

Select a peer i proportional to its fitness. This is done in the following way: generate a uniformly random number r from [0,1]; sort the peers according to their fitness values; start summing up the fitness values of the (sorted) peers, and denote this (partial) sum by s; normalize s; select the peer i at which s is greater than r.

Select another peer j uniformly at random and change its strategy to the strategy of peer i.

Apply mutation on peer j.

Reset $f_j=0$.

Satisficing updating

In this decentralized selection rule we assume that all the peers keep record on their own fitness value from the previous R rounds. This value is $\hat{f}_i$, where $i=1,...,n$. The general idea of satisficing was proposed in [19]. It captures the idea that individuals may be satisfied with an internally calculated threshold rather than comparing themselves to others.

Repeat K times:

Select peer j uniformly at random

If $f_j\geq\hat{f}_j$ holds, then peer j keeps its current strategy, otherwise it switches to the opposite one.

Apply mutation on peer j.

4 Experiments

We compare the strategies tested in the simulations using the DSA approach previously discussed. Firstly, every strategy (protocol variant) is evaluated for Performance (based on average download time) when it is in a population composed entirely of the same strategy. This involves a realistic BitTorrent simulation (BitTorrent game) in which each peer attempts to download a file.

Then every possible pair of strategies are pitted against each other in a tournament by creating a population composed of half of each and executing the BitTorrent simulation. At the end of the simulation run the best performing strategy in terms of average performance is deemed to have "won" the tournament. A robustness measure is calculated for each protocol by averaging wins over all tournaments against all other protocols. Hence, a robustness value of 1 indicates a protocol wins against all others whereas a value of 0 indicates it loses against all others.

Finally, we apply the evolutionary extension by modifying the above tournament process in the following way. During a tournament, periodically, peers may change their strategy to their opponents strategy using one of the evolutionary algorithms described above. At the end of a simulation run the best performing strategy in terms of evolutionary success is deemed to have "won" the tournament. Wins are classified as "weak" or "strong" (see below for definitions). A strong and weak evolutionary robustness measure is calculated for each protocol by averaging over all tournaments.

In all cases simulations involved 10 independent runs starting with different pseudo-random number seeds and averages were calculated. This is necessary because the BitTorrent game simulator involves several stochastic elements.

4.1 Actualization of BitTorrent strategies

We selected a subset of the BitTorrent strategies tested in [6] covering the main interesting behavioral variants. The dimensions and possible values for each dimension are described below.

Stranger policy: Three kinds of policies are used here:

- Periodic: Give resources to strangers periodically.
- When needed: Only give resources to strangers when do not have enough regular partners.
- Defect: Never give resources to strangers.

Number of strangers can be either 0, 1 or 2.

Selection function: This depends on the Candidate list, Ranking function and the number of peers selected:

- For the *Candidate list* we use the BitTorrent's default *TFT*, in which a peer only places those peers in the candidate list who reciprocated to it in the last round.
- For the *Ranking function* we use all six actualizations: *Sort Fastest*, *Sort Slowest*, *Sort Based on upload bandwidth proximity* (called '*Birds*'), *Sort Adaptive* (ranks peers in order of proximity to an aspiration level, which is adaptive and changes based on a peer's evaluation of its performance), *Sort Loyal* (ranks peers in order of those who have cooperated with the peer for the longest durations), and *Random*.
- the number of top k peers selected after applying the raking function can be from the range [0,4]. Note that the case k=4 is in line with the default BitTorrent parameter for the number of partners in a peer's unchoking slots.

Resource allocation: Two allocation methods are tested:

- *Equal split* gives all selected peers equal resources (upload bandwidth).
- *Freeride* gives nothing to partners.

Altogether, the above dimensions specify a space of 540 different protocol variants.

4.2 Strong and weak evolutionary robustness

Assuming that $R=R_T$, $K=N$ and w=1 in our algorithm, the robustness value of a protocol P is calculated in the following way. For each run, we compare the average fitness value of P with the average fitness of the other protocol. If the average fitness of P is greater than the average fitness of the other protocol, we mark it as a '*Win*' for *P,* otherwise we mark it as a '*Loss*' for P. The robustness value for P is calculated by number of games that it wins against all opponents in all runs divided by the total number of games that it plays, which is constant for all protocols.

This single robustness measure is sufficient when no evolutionary algorithm is applied. However, when applying an evolutionary algorithm this approach does not capture the possible dynamics of strategy change over time. We therefore introduce two new robustness measures for evaluating a protocol variant within an evolutionary algorithm: *weak evolutionary robustness* and *strong evolutionary robustness*.

Having evaluated the selection rule in Step 3 in the Algorithm explained in Section 3, we denote the number of peers associated with strategy A and B as N_A and N_B, respectively.

Based on this fact, we define two kinds of *'win'*:

- If $N_A > N_B$ holds in each and every Rth round of the algorithm, then we mark this as a '*strong win*' for protocol A.
- If $N_A \geq N_B$ holds at some Rth round of the algorithm, and $\sum N_A \geq \sum N_B$, then we mark this as a '*weak win*' for protocol A.

The *strong evolutionary robustness* (and the *weak evolutionary robustness*) value of protocol P is calculated by the number of games that it strongly wins (respectively weakly wins) against all opponents in all runs divided by the total number of games that it plays. For illustration of the definitions strong and weak evolutionary robustness see Figure 1. A protocol is strongly robust against another one if it never becomes a minority in the population during the simulation run. If the protocol was in the majority for most of the time, but not in all rounds, then it is weakly robust against the competing protocol variant[1].

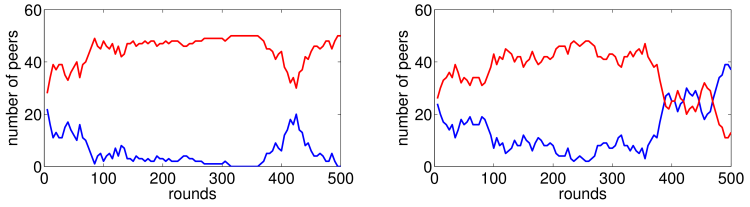

Figure 1

Illustration of strong evolutionary robustness (left) and weak evolutionary robustness (right) of a protocol denoted with red line

Weak and strong evolutionary robustness measures differ from the standard non-evolutionary robustness measure because they do not measure relative performance between protocol variants directly, but rather measure their replication success based on the outcome of an evolutionary algorithm.

5 Results

We have extended the cycle-based simulator used in [6] with the selection rules (listed in Section 3.1) and with the ability to measure the strong and weak evolutionary robustness of protocols (which were introduced in Section 4.2). In the following we use the terminologies introduced in Section 4.1.

[1] The situation where both protocols stayed evenly matched throughout all rounds was not found to occur in practice.

The simulation models a single swarm where peers share the same file. Time in this model consists of *discrete rounds*. For peer discovery we assume that all peers can connect to each other. In every round each peer is fired in random order and engages in connection and data transfer activities. It is assumed that all peers always have data that others are interested in. In each round, a peer:

- decides to upload to (maximum) 4 peers which are selected by the Selection function;
- uses its Resource allocation policy to decide how much to give to each of the selected partners;
- decides whether to cooperate or not with strangers, which is based on its Stranger policy.

Each peer maintains a short history of actions by others. This is implemented as a list, which has typically a length of 10-15 elements. At the same time, a peer also has some rate of requesting services from other peers that depends on specific actualizations. This means that different protocols will request services at different rates. This is how optimistic unchoking is mapped in our scenario. For example, a freeriding protocol will request a service with a partner every round and offer no bandwidth. If the chosen partner is a sucker (i.e., it accepts those protocols which provide zero bandwidth), then this partnership will be maintained further on.

Each simulation experiment was run with 50 peers and for 500 rounds in total. The peers' upload bandwidths are initialized using the bandwidth distribution provided by [2]. This was done by deriving uniformly at random samples from this dataset to assign to the peers in the simulations, hence preserving the distribution. The distribution represents real download rates that were empirically collected. Download is assumed to be infinite[2].

In the evolutionary approaches tested below we used the parameters: number of rounds R_T=500, number of peers R=50, selective pressure K=5, mutation rate m=0.01, and intensity of selection w=1 as defaults. Note that we also used other parameter setups and found no significant differences from the results obtained using the defaults.

Given the abstraction level of the simulator only the above parameters are required. There is no explicit representation of pieces or seeding/leeching behavior. Hence, the focus is on the effect of strategy interactions on data sharing[3].

[2] Since upload bandwidth is the main constraint in file sharing systems this assumption does not significantly effect results.

[3] A previous version of the simulator was validated against an actual BitTorrent client implementation and experiments [6], which gives us some confidence in the validity of the results.

5.1 Without Evolution

Firstly, we calculated performance and robustness values for all our 540 protocol variants *without* applying evolution - see Figure 2. Note that Performance is normalized over the entire protocol design space. These results serve as a baseline for comparison with the evolutionary approaches. Also, we were interested to see the effect of selecting only a subset of potential protocol variants from [6]. We categorize those protocols which never share as freeriders.

We found that the ranking of the protocols is comparable to those previously obtained. This result is non-trivial and reassuring. This is because it is the nature of the DSA approach that the performance and robustness of any given strategy is only calculated relative to the other strategies in the design space. This means that results obtained for one space of strategies can not be generalized to either a subset or superset of strategies without testing. Hence our reproduction of results for our chosen subset allows us to be more confident that the previous results obtain were not merely the result of an artifact of large the design space chosen there.

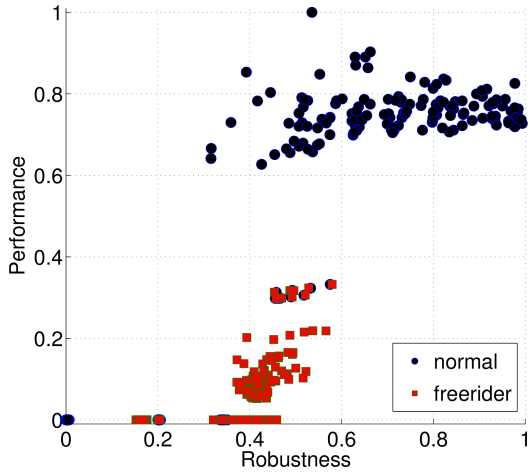

Figure 2

Robustness against Performance using 540 protocol variants without apply evolution

5.2 Tournament selection[4]

We performed evolutionary simulations for all protocol variants using the tournament selection approach. We then analyzed these results by comparing them to the Performance and Robustness obtained from the previous non-evolutionary

[4] Note that the term "tournament selection" should not be conflated with the "tournament" nature of the DSA approach. These refer to two different kinds of tournament occurring at two different levels.

simulations. As previously stated (in Section 4.2), each protocol is measured along two new measures - strong evolutionary robustness and weak evolutionary robustness.

Figure 3 shows the results we got for the Robustness test. We can immediately see that strategies cannot be quantified using weak evolutionary robustness, all the protocol variants have roughly the same value. On the other hand, strong evolutionary robustness correlates with the previously obtained non-evolutionary robustness. The Pearson's correlation coefficient of robustness and strong evolutionary robustness is 0.939.

This result indicates that evolution based on tournament selection effectively reproduces the non-evolutionary DSA approach so long as strong evolutionary robustness is used as a measure for winning. Or to put this another way, the non-evolutionary DSA approach is a good predictor of strong evolutionary robustness in this evolutionary setting. This is a non-trivial result because strong evolutionary robustness measures the outcome of an evolutionary processes over time, whereas non-evolutionary robustness is based on relative performance in one-shot interactions.

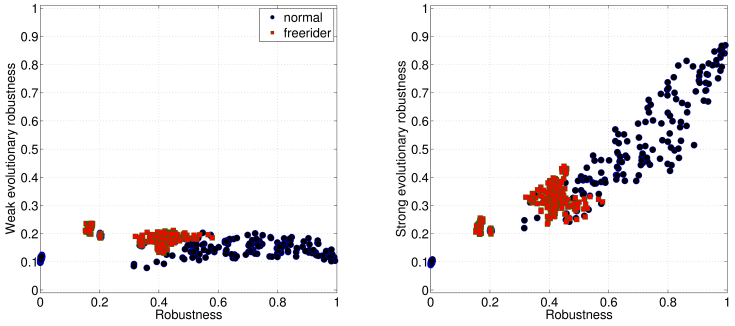

Figure 3

Tournament selection – weak evolutionary robustness and strong evolutionary robustness compared to non-evolutionary robustness

Figure 4 shows weak and strong evolutionary robustness against performance. Note that performance is calculated identically for both non-evolutionary and evolutionary approaches. Hence, performance values are identical to those given in Figure 2. As we would expect, from the results given in Figure 3, we see that weak evolutionary robustness does not distinguish between performance whereas strong evolutionary robustness reproduces the non-evolutionary results and is directly comparable with Figure 2. The Pearson's correlation coefficient of performance and strong evolutionary robustness is 0.761.

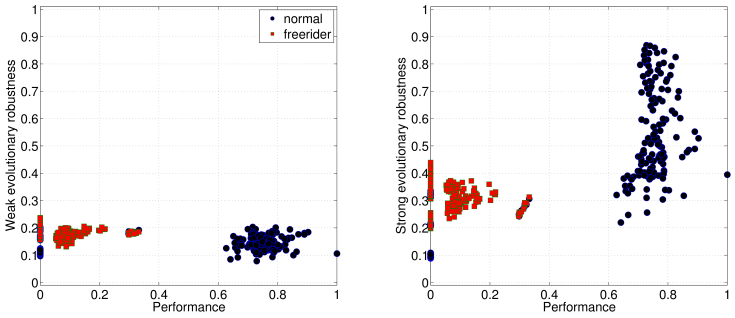

Figure 4

Tournament selection – performance

5.3 Death-birth selection, Birth-death selection

For both Death-birth and Birth-death selection we obtain almost identical results to those for Tournament selection. Figure 5 shows the correlation between the weak and strong evolutionary robustness measures of the different protocol variants we found using Tournament and Birth-death selection. As we can see the weak evolutionary robustness values are very strongly correlated. In the case of strong robustness we notice that some strategies can have significantly higher value under Birth-death selection. Closer inspection of these particular variants indicated they used the Periodic or When-needed stranger policy, Sort fastest ranking function, 1 or 2 regular and optimistic unchoke slots and Equal split as resource allocation.

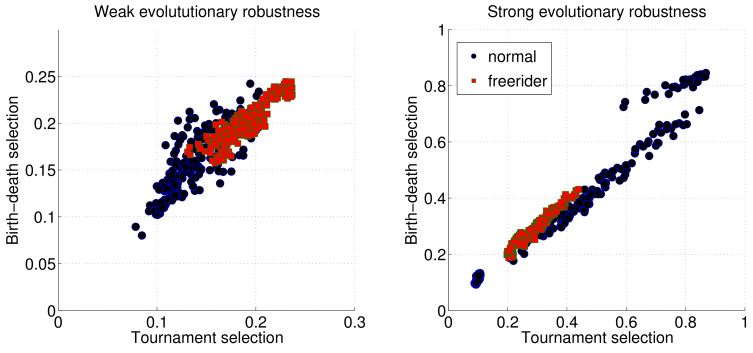

Figure 5

Comparison of Tournament and Birth-death selection.

Interestingly, the Death-birth selection mechanism produced very much the same ranking as Birth-death selection. Namely, we got 0.987 and 0.998 Pearson's correlation coefficients for weak and strong evolutionary robustness, respectively.

We can conclude that these evolutionary algorithms also reproduce the non-evolutionary results when the strong evolutionary robustness measure is used. This indicates that different evolutionary algorithms based on fitness comparisons are predicted by the non-evolutionary DSA approach.

5.4 Satisficing selection

This selection mechanism differs from the above tested ones as it is using only local information. This means that no fitness comparisons are made between peers but rather individual peers assess their own performance against an internal threshold or target.

As can be seen in Figure 6 and 7, neither weak nor strong evolutionary robustness make any significant distinction between the different protocol variants. In this case we cannot even distinguish between freeriders and normal strategies.

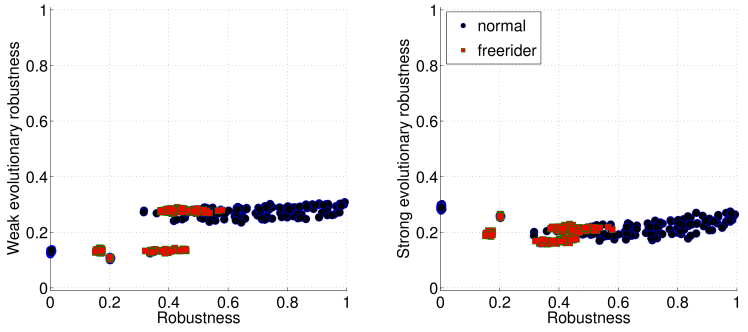

Figure 6

Satisficing selection – robustness

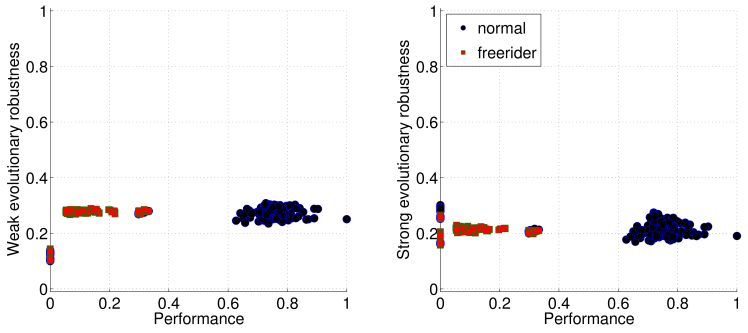

Figure 7

Satisficing selection – performance

In general outcomes appear to be little more than random noise. Since there are four possible outcomes from a tournament we would expect uniformly random outcomes to produce weak and strong evolutionary robustness values around 0.25.

Note, however, a small cluster of freeriders can be seen to have noticeably lower weak evolutionary robustness values around 0.4 on the *x*-axis (Figure 6, left hand side). These appear to be a special case of very poorly performing protocols, as can be seen in Figure 7 (left hand side) where this cluster is bunched at 0.0 on the *x*-axis.

More interestingly, a small cluster of 72 non-freerider variants can be seen in Figure 6 (right hand side) with near-zero robustness, but relatively high levels of strong evolutionary robustness (around 0.3). This means that these variants which have almost zero non-evolutionary robustness do much better in an evolutionary setting. These protocols do not communicate with strangers (thus they are unable to bootstrap in an homogeneous environment, i.e., producing zero performance), use Equal split as resource allocation and have $k>0$ regular partners. Closer inspection of the simulation results revealed that these particular variants *always* dominate in scenarios where they are paired with freeriders. This does not imply, though, that these protocols do better than freeriders. Due to the local nature of the calculation of the satisfaction threshold this only means that variants using these kind of protocols do gradually better than in the previous round, whereas the competing freerider protocol variant' performance varies up and down.

Overall we can see that the non-evolutionary DSA approach does not predict the outcome of the satisficing approach. However, there are *many* ways to implement a satisficing approach and it would be interesting to explore other implementations to identify conditions under which (if any) results would converge to the previous DSA results. For example, some satisficing approaches utilize an adaptive satisfaction threshold that may increase above over all previous performances obtained through the application of noise (a "trembling hand effect") [20]. Alternatively a minimum as well as maximum threshold could be employed.

Conclusions

We developed a limited evolutionary extension to the Design Space Analysis approach [6]. We applied this extended approach to an exploration of BitTorrent protocol variants capturing the possibility of dynamic changes in protocol variants over time within pairwise tournaments between protocols. We found that the results obtained were broadly consistent, in most cases, with those previously obtained thus increasing our confidence in previous results as a predictor for this form of limited evolution.

The subset of protocol variants that we selected ranked similarly to the larger design space results. This is a non-trivial observation since results form a given design space do not necessarily generalize to a subspace. This is due to the co-evolutionary nature of open peer-to-peer systems - where the utility of each peer is dependent on the dynamic composition of protocol variants in the population as a whole rather than being related to a fixed and equal partition.

We found that applying a satisficing approach did *not* reproduce the previous results and in fact, as implemented here, produced almost random outcomes. However, some of the results point towards the kinds of alternative satisficing approaches that might produce comparable outcomes. It is of interest how different satisficing approaches behave because it could be argued that such a procedure may capture the kinds of user behavior that leads to a change of protocol variants over time - since users tend to have access only to local information on protocol performance.

The work here limits coevolution to all pairs of variants. To fully understand co-evolutionary dynamics it would be necessary to allow for many protocol variants from the design space to exist in the population simultaneously. However, previous work has demonstrated the difficulty in analyzing the results of evolution applied to populations with design spaces as low as six variants [16]. Our aim in limiting evolution in this way is as a step *towards* modeling and understanding co-evolutionary processes in large design spaces while linking back to previous results. This requires careful experimental design and analysis to avoid a combinatorial explosion and to filter for noise which plays a large role in protocol pairings and can be magnified by coevolution.

Future work may develop coevolutionary algorithms that can be applied to a large design space within a single population by grounding the algorithms in a theory of user and developer behavior. Developers modify and release new protocol variants and users decide if to download and use them. It is not beyond current approaches to model this process in a coevolutionary simulation, but more detailed user and developer models would need to be formulated.

Acknowledgement

This work was partially supported by the European Union and the European Social Fund through project FuturICT.hu (grant no.: TAMOP-4.2.2.C-11/1/KONV-2012-0013). T. Vinkó was supported by the Bolyai Scholarship of the Hungarian Academy of Sciences.

References

[1] B. Cohen, Incentives build robustness in BitTorrent, In Proceedings of Workshop on Economics of Peer-to-Peer Systems, 68-72, 2003

[2] M. Piatek, T. Isdal, T. Anderson, A. Krishnamurthy, and A. Venkataramani. Do incentives build robustness in BitTorrent. In Proceedings of NSDI, vol. 7., 2007

[3] T. Locher, P. Moor, S. Schmid, and R. Wattenhofer. Free riding in BitTorrent is cheap. In Proc. Workshop on Hot Topics in Networks (HotNets), 85-90. 2006

[4] U.W. Khan, and U. Saif, BitTorrent for the less privileged. In Proceedings of the 10th ACM Workshop on Hot Topics in Networks (HotNets-X), 2011

[5] N. Liogkas, R. Nelson, E. Kohler, and L. Zhang, Exploiting BitTorrent For Fun (But Not Profit), In Proc. of IPTPS, 2006

[6] R. Rahman, T. Vinkó, D. Hales, J. Pouwelse, and H. Sips, Design Space Analysis for Modeling Incentives in Distributed Systems. In Proceedings of the ACM SIGCOMM 2011 Conference (SIGCOMM'11) 182-193, 2011

[7] R. Axelrod, The Evolution of Cooperation. Basic Books, 1984

[8] R. Dash, N. Jennings, and D. Parkes. Computational-mechanism design: A call to arms. IEEE Intelligent Systems, 18:40–47, 2003

[9] J. Feigenbaum and S. Shenker. Distributed Algorithmic Mechanism Design: Recent Results and Future Directions. In ACM DIALM, 2002

[10] R. Mahajan, M. Rodrig, D. Wetherall, and J. Zahorjan, Experiences applying game theory to system design. In Proceedings of the ACM SIGCOMM workshop on Practice and theory of incentives in networked systems (PINS '04). ACM, New York, NY, USA, 183-190, 2004

[11] C. Buragohain, D. Agrawal, and S. Suri. A game theoretic framework for incentives in P2P systems. In Proceedings of IEEE P2P, 2003

[12] M. Feldman, K. Lai, I. Stoica, and J. Chuang. Robust incentive techniques for peer-to-peer networks. In Proceedings of ACM EC, 102–111, 2004

[13] A. Bharambe, C. Herley, and V. Padmanabhan. Analyzing and improving a BitTorrent network's performance mechanisms. In Proceedings of INFOCOM, 1-12, 2006

[14] M. Meulpolder, J. Pouwelse, D. Epema, and H. Sips. BarterCast: A practical approach to prevent lazy freeriding in P2P networks. In Proceedings of IEEE IPDPS, 2009

[15] J. Mol, J. Pouwelse, M. Meulpolder, D. Epema, and H. Sips. Give-to-get: Free-riding-resilient video-on-demand in p2p systems. In Proceedings of SPIE/ACM MMCN, 2008

[16] X. Jin, Y.-K. Kwok, J. Deng, Variegated competing peer-to-peer systems with selfish peers, Computer Networks, (24)313-330, 2014

[17] R.L. Riolo, M.D. Cohen, and R. Axelrod, Evolution of cooperation without reciprocity. Nature, 414(6862), 441-443, 2001

[18] H. Ohtsuki, C. Hauert, E. Lieberman, and M.A. Nowak, A simple rule for the evolution of cooperation on graphs and social networks. Nature, 441(7092), 502-505, 2006

[19] H. A. Simon, Models of Bounded Rationality, Vol. 1, MIT Press, Boston, 1984

[20] C.P. Roca and D. Helbing, Emergence of social cohesion in a model society of greedy, mobile individuals, PNAS 108 (28) 11370-11374, 2011