On the Evaluation of Services Selection Algorithms in Multi-Service P2P Grids

Álvaro Coêlho, Francisco Brasileiro

Universidade Federal de Campina Grande
Departamento de Sistemas e Computação
Laboratório de Sistemas Distribuídos
Av. Aprígio Veloso, s/n, Bloco CO
58.109-970, Campina Grande - PB, Brazil
Emails: degas@lsd.ufcg.edu.br, fubica@dsc.ufcg.edu.br

Abstract—It has been shown that the use of a reciprocation mechanism in peer-to-peer grid systems which provide multiple services to their users is an efficient way to prevent free-riding and, at the same time, to promote the clustering of peers that have mutually profitable interactions. However, when peers are subject to resource limitations, they may be unable to offer all possible services and shall select a subset of services to offer. Previous work showed that the overall profitability of a peer is strongly dependent on the set of services it offers. Thus, the use of an appropriate services selection algorithm is crucial to yield better profitability to peers. Clearly, evaluating the efficiency of services selection algorithms is an important aspect in the search for suitable solutions for this problem. Unfortunately, due to the complexity and inherent non-determinism of the system, it is normally intractable to compute optimal solutions even for small systems. This renders the task of evaluating the performance of practical heuristic-based algorithms difficult. This work aims to fill in this gap by providing a cheaper evaluation method. The methodology we propose maps the services selection problem into the well-known knapsack problem, making brute force techniques affordable for reasonably large systems. Then, by immersing the algorithms under evaluation on a similar setting, it is possible to assess their efficiency compared to an optimal solution. We show how the methodology can be used by evaluating two services selection heuristics.

I. INTRODUCTION

Nowadays, peer-to-peer (P2P) systems are becoming an interesting way to obtain files, storage, processing and many other computational resources with high levels of availability. In such systems, participants play both the role of a client and that of a server [1]. Peers who are acting as servers must schedule the use of their resources in order to provide services to the other peers that are acting as clients at that time.

The scheduling of resources is currently done using two possible approaches: market-based and sharing-based [2]. Market-based approaches have been used in many systems [3], [4], [5]. Although market-based solutions give more strict guarantees on the quality of service provided, the nature of this approach implies in higher transaction costs for peers. This occurs because this kind of solution relies on the existence of contract negotiations, a system for distributing currency, banking services, auditing, and accurate pricing. On the other hand, no currency or any trustworthy central institution are used in sharing systems. Peers exchange resources based on a reciprocation scheme that may use some sort of incentive mechanism. This kind of mechanism normally uses only loosely structured information, which is much simpler to obtain. Also, peers can use social mechanisms for monitoring and enforcement. Therefore, this kind of solution leads to lower marginal transaction costs [6].

The Network of Favors (NoF) [7] is a mechanism that provides incentive for the donation of spare resources in a P2P computational grid. The idea is that in a P2P computational grid, peers offer their idle processing power to other peers that have demand for them. In exchange, a peer that has donated idle cycles in the past, expects to be able to use other peers’ excess computational power when its demand cannot be fully served by its local resources. In effect, peers exchange computing “favors”. Under the NoF scheme, each peer maintains, for each other peer with which it has interacted in the past, a balance of its direct past interactions with the other peers. Whenever the idle resources of a peer are contested by more than one peer, these resources are allocated proportionally to the peer’s debt with the requesting peers. This simple mechanism has been proved to inhibit free-riding in the system, at the same time that makes in the best interest of each peer to donate as much resources as possible to the system.

The original NoF mechanism was proposed for a system in which a single service (raw processing) was exchanged [8]. Some additional considerations must be taken into account when peers are exchanging multiple services, as software execution, file storage, database processing, etc.. In this setting, there are at least two new issues that need to be considered: i) peers may exchange a service of one kind for a service of another kind, and ii) peers may value services differently [9]. As a consequence, the links that are formed between peers
are based not only on the peers’ behavior but also on the profitability of their interactions. A slightly modified version of the NoF for P2P computational grids offering multiple services has been proposed [9]. By using this mechanism, peers are able to autonomously identify which are the other peers with whom mutually profitable interactions are possible. Obviously, since interactions with free riders are always unprofitable, free riders are also marginalized, just like in the single service case.

In view of the multitude of services that can be offered, and the fact that peers have limited resources, normally, not all services can be offered by all peers. As a result, each peer has to choose a subset of services to offer from the set of all possible services available. Since each possible subset has a specific cost and returns different utility, the selection of this set of services has a direct impact on the profitability that a peer can extract from the grid [10].

Some services selection algorithms based in heuristics have been proposed; their performance can be substantially better than that attained when services are selected randomly [11]. Nevertheless, it is not clear how close to an optimal choice these algorithms are, since the complexity of executing a brute force algorithm rules out the possibility of computing the optimal choice at an affordable cost.

In this paper we try to circumvent this limitation by proposing a practical methodology to evaluate heuristic-based services selection algorithms. We introduce some simplifications that allow the services selection problem to be mapped into an instance of the well-known knapsack problem. Moreover, we add the assumption that the algorithm has access to privileged information that is not normally available to it. Then, a referential brute force algorithm can be used to find out optimal solutions for this simplified problem, considering reasonably large systems. The performance of heuristic-based algorithms can be assessed by executing them in the same setting, however, without access to the privileged information.

The remainder of the paper is organized as follows. Section II gives a formal description of the services selection problem. In section III we present a brief description of the knapsack problem, followed by a discussion on how a simplified version of the services selection problem can be mapped onto the knapsack problem. Then, in Section IV, we explain how a referential algorithm can be used to evaluate heuristic-based services selection algorithms. We describe some heuristics to be evaluated and experiments using the proposed approach. In Section V, we discuss related work. Finally, Section VI closes the paper with our concluding remarks and perspectives of future work.

II. THE SERVICES SELECTION PROBLEM

We consider a P2P grid in which each peer consumes and provides several different services, and each service is likely to have different costs for different peers [10]. Services also provide different utilities to the different clients that use them. Moreover, peers have limited resources and may not be able to provide all possible services. Thus, peers need to select a service portfolio to provide, within the limits of their resources.

A. System Model

We assume that the system is formed by \( N \) peers that share their spare resources. The tuple \( S = (s_1, s_2, ..., s_n) \) represents the list of all possible services that can be provided by the peers. Peers have different costs to provide services, however, for the sake of simplicity, we assume that these costs are constant over time. We use the notation \( c_{p,s} \) to represent the cost for peer \( p \) to make available service \( s \). We model the resource limitations of peers by imposing a maximum budget for each peer to provide services during a given period of time. We consider that this period of time is slotted in turns \( t_1, t_2, ..., t_T \), and the duration of each turn \( t_i \) is large enough to allow management decisions to be performed and the exchange of services among peers. We consider that peer \( p \) has a budget \( B_p \) to set and maintain its services for the whole period of \( T \) turns. Again, for simplicity, we assume that the budget is equally distributed in the \( T \) turns, i.e. the maximum budget that can be used by \( p \) in any turn is \( B_p/T \).

At the beginning of each turn, every peer must select a service portfolio that it will provide during that turn. The tuple \( P_{p,t} = (a_{p,1,t}, a_{p,2,t}, ..., a_{p,n,t}) \), with \( a_{p,s,t} \in \{0,1\}, \forall p, s, t | 1 \leq p \leq N, 1 \leq s \leq n, \text{ and,} 1 \leq t \leq T \), represents, for turn \( t \), the availability of services in the portfolio offered by peer \( p \). Service \( s \) is offered by peer \( p \) in turn \( t \) only if \( a_{p,s,t} = 1 \).

We also consider that each peer has a typical favor request profile, which is represented by the subset of required services and the proportions among these services. Hereafter, we assume that the average proportion of services requested by peer \( p \) in each turn is represented by the tuple \( F_p = (f_{p,1}, f_{p,2}, ..., f_{p,n}) \), where \( 0 \leq f_{p,s} \leq 1, \forall p, s | 1 \leq p \leq N, 1 \leq s \leq n, \text{ and, } \sum_{s=1}^{n} f_{p,s} = 1 \).

We consider that peers will use an incentive mechanism, such as the NoF [7], to guide the scheduling of their resources. Therefore, when a peer \( p \) provides a given service \( s \) to other peers, it expects to be reciprocated in the future by consuming services provided by these other peers. Thus, in each turn, \( p \) will receive a certain future utility derived from the services that it has provided. Let \( \lambda_{p,s,t} \) represent the future utility that service \( s \) yields to \( p \) by the fact that \( s \) was offered by \( p \) in turn \( t \). Then, the profit yield by the offering of service \( s \) by \( p \) in turn \( t \) is given by:

\[
\pi_{p,s,t} = \lambda_{p,s,t} - c_{p,s}.
\]

B. Problem Statement

A solution to the Services Selection problem aims at maximizing the profit of the peer, by choosing the best subset of services subject to the peer’s budget limitation. Formally, the problem can be formulated for each peer \( p \) as:

\[
\text{Maximize } \sum_{j=1}^{n} \sum_{t=1}^{T} \pi_{p,j,t} \cdot a_{p,j,t}
\]
subject to

\[ \sum_{j=1}^{n} c_{p,j} \cdot a_{p,j,t} \leq B_p / T. \]

III. MAPPING THE SERVICES SELECTION PROBLEM INTO THE KNAPSACK PROBLEM

The knapsack problem has been intensively studied since Dantzig’s seminal work [12], in part for immediate applications, in part for theoretical reasons. Several variations of the problem have since been proposed and studied. In common, all members of the knapsack problem family belong to the complexity class $NP – Complete$ [13].

The problem was first introduced as follows. A vendor uses a knapsack to store items to be sold. Each item has a value and a weight, and the knapsack supports a maximum load. The question is, which is the set of items that maximizes the total value into the knapsack, not exceeding its maximum load?

Formally, the problem can be described as follows. Given $n$ kinds of items, $(i_1, ..., i_n)$, each item $i_j$ has a value $p_j$ and a weight $w_j$. The total weight supported by the knapsack is $c$. There are three vectors: $W$ and $P$, which define the items weights and values, respectively, and $X$, which indicates the amount of units of each kind of item that are inside the knapsack.

The problem can be formulated as:

Maximize $\sum_{j=1}^{n} p_j \cdot x_j$

subject to

$\sum_{j=1}^{n} w_j \cdot x_j \leq c$.

The 0 – 1 knapsack problem is the most well-known variation of the original problem and is formulated in an identical way, with the extra constraint that $x_j \in \{0, 1\}$.

The similarity between the services selection problem and the 0 – 1 knapsack problem is quite obvious. The direct mapping between item weights and service costs, item values and profit yield by services, and the knapsack capacity and the peer’s budget allow us to say that the computational complexity for services selection problem is at least equal to that of the knapsack problem. Since the knapsack problem is $NP – Complete$, it follows that the services selection problem is $NP – Complete$ too.

However, it is important to point out that, unlike the knapsack problem, where the items values are known, in the services selection problem there is an important constraint: the future utility from a service depends on several factors that escape from the control and knowledge of the peer which is offering services. These factors include, among other things, the demand and offer patterns of the other peers, as well as the cost incurred and the value yield by the services that are provided and consumed by the other peers. Also, the 0 – 1 knapsack problem is defined for a single knapsack and for a single selection of the contents of the knapsack. In contrast, the services selection problem involves several autonomous “knapsacks” (one for each of the $N$ peers in the system), selected independently by each peer, considering a sequence of “knapsacks” (for each turn $t$).

To allow the mapping of the services selection problem into the knapsack problem and, therefore, allow us to use solutions to the knapsack problem to solve the services selection problem, we introduce some simplifications. First, since peers are selfish agents aiming at obtaining individual advantages, we can consider the problem of how each of them can maximize its own profit, i.e. the fact that other peers are also trying to maximize their profit is encapsulated in the future utility that a peer receives for providing a given service at a given turn. Also, the system is augmented with accurate knowledge about the future utilities attained. Finally, we solve one independent knapsack problem for each turn $t$.

There are many different strategies to deal with all kinds of knapsack problems. Some of them are useful to all, or almost all knapsack problem variations, whether others are limited to specific ones. Since the problem is $NP – Complete$, all strategies make a choice between performance and accuracy. Considering classical approaches, greedy and genetic algorithms are executed in polynomial time (greedy is faster although genetics is more accurate [13]) and dynamic programming uses an exponential or pseudo-polynomial time$^1$.

In common, these strategies search only for results considered “good” instead of “the best” [14] [15], [16], [17], [18], [19], [20].

However, the search for an optimal solution is still necessary for two reasons. Firstly, because in some situations it is imperative to find the best solution. In this case, it is possible to wait for the necessary time to compute an off-line solution. Normally, parallel algorithms are used in order to find the optimal solution in a reasonable time [21] [22]. The second reason is the validation of heuristic-based algorithms: polynomial algorithms find non-optimal solutions using heuristics. So, it is necessary to evaluate their efficiency, i.e. how near is a solution compared to the optimal one. In this sense, a common approach to find the optimal solution is to use an exhaustive search.

Unfortunately, even if the complexity to find a solution can be dealt with similarly as it has been the case for the knapsack problem, algorithms that solve the simplified version of the services selection problem are not useful in practice. This is because the knowledge artificially added is not available in real systems. Nevertheless, as discussed above, optimal solutions to the simplified problem are useful to establish the performance limits that can be achieved by any solution to the services selection problem executed in the same setting. In particular, heuristic-based algorithms that solve the simplified problem without using the added information can be useful in practice.

---

$^1$There is an implementation that creates a table having $n$ rows, $n$ being the number of items, and $c$ columns, $c$ the knapsack capacity. This algorithm runs in $O(n \cdot c)$. However, the $c$ value can imposes a high spacial complexity.
and can have their performance compared to the unrealistic referential algorithm that computes the optimal solution using the added information, possibly using a brute force method.

IV. EVALUATING HEURISTICS

In this section we show how to use a referential algorithm to evaluate heuristic-based algorithms that solve the services selection problem. We first describe two of such algorithms, then we discuss how the additional information required to run the referential algorithm can be computed. This is followed by a discussion on how we can analytically calculate the performance of an algorithm that makes a random selection of the provided services by a peer. This can be seen as a kind of practical lower bound on the performance of services selection algorithms. Finally, we present a performance evaluation of the proposed heuristics.

A. Heuristics to Solve the Services Selection Problem

We defined two services selection algorithms to be evaluated using the approach we propose in this paper. The first one, called Cost-based, is inspired in well-known solutions to the knapsack problem. It uses a greedy approach, but using only the items’ weight (i.e. the services’ cost) instead of the items’ density (i.e. the services’ future utility divided by their costs), because the value (i.e. the services’ future utility) is unknown.

The second algorithm, called Reciprocation-based, is inspired in a version of the Portfolio Selection Problem. This problem was originally proposed by Markowitz and has received considerable attention, being studied by researchers working in the areas of economics, mathematics, operational research and computer science.

The problem is described as an investor who has access to a set of stocks, with each one returning a profitability, with an associated risk. A portfolio is a decision about how much of the investor’s budget will be used in each stock. The goal is to conciliate risks and profits. The profitability is unknown, but it follows a normal distribution. Markovitz’s analysis considers historical data to define the expected profitability, given by the average, and the risk, given by the standard deviation. The question to answer is then: given stocks with historical profits, which portfolio, considering some acceptable limit on the risk, maximizes the expected profitability?

Although the future stock profitability is unknown, one can consider the portfolio selection problem with some predictive information when it is assumed that the investor knows the order of the expected profitability yield by the stocks. This problem is known as the Portfolio Selection with Order of Expected Returns. Considering this, the portfolio selection problem is defined as:

Maximize

\[
(1 - w) \cdot \sum_{i=1}^{n} R_i \cdot x_i - w \cdot \sum_{i=1}^{n} \sum_{j=1}^{n} \sigma_{ij} \cdot x_i \cdot x_j
\]

Subject to

\[
\sum_{i=1}^{n} x_i = 1,
\]

\[
R_i \geq R_{i+1}, i = 1...n - 1,
\]

\[
a_i \leq R_i \leq b_i, i = 1...n,
\]

\[
0 \leq x_i \leq 1, i = 1...n,
\]

where \( n \) is the number of stocks, \( x_i \) is the proportion of the budget that was invested in the stock \( i \), \( R_i \) is the expected return from stock \( i \), \( \sigma_{ij} \) is the covariance between the historical profits from stocks \( i \) and \( j \), and \( w \) is the risk-aversion factor, with \( 0 \leq w \leq 1 \).

Note that the lower is the covariance between stock profits, the lower is the risk for an investor who selects both stocks in her portfolio. Furthermore, \( w \) models the investor profile: if an investor is more aggressive (\( w \rightarrow 0 \)) she will give little consideration to the covariance; on the other hand, if the investor is more conservative (\( w \rightarrow 1 \)) she pays more attention to the risk.

In a P2P grid as we consider, peers can also make an estimation about the order of services according to their expected future utility. This is possible because the more reciprocal a peer who has consumed services in the past, the more profitable is the service that was offered to it. For instance, if the system uses the NoP scheme, peers can use the balance that it maintains for every other peer in the system to know with which peers the past interactions were more profitable and select services according to this expected order of utility.

The Reciprocation-based heuristic is inspired in the Portfolio Selection with Expected Order of Returns. It selects services according to the expected order of reciprocation of the known peers.

B. Instantiating the Simplified Services Selection Problem

To instantiate the simplified services selection problem we need to provide, for the peer \( p \), the values of the future utility yield by each service \( s \) in each turn \( t \). A possible way to calculate these values is to use an economical model such as the one presented in [11]. Basically, this model accounts the cost and the utility attained by a peer in any instant of time. Costs are expressed as we modeled in this paper, and utilities are expressed as a multiple of the cost that would be incurred by a peer to provide the service consumed. That is to say, if peer \( p \) consumes \( x \) units of service \( s \), then this yields a utility to \( p \) that is equal to \( x \cdot \mu_{p,s} \cdot c_{p,s} \), where \( \mu_{p,s} \) is a profitability factor for the service \( s \) consumed by peer \( p \).

Unfortunately, anticipating how much of service \( s \) a peer \( p \) will receive in turn \( t \) is not very practical. This occurs because, due to the reciprocation scheme, the service that \( p \) receives in a turn \( t \) from a peer \( q \) is normally a consequence of \( p \) having offered some service to \( q \) in the past. So, choices
made by peer \( p \) in a given instant influence future scheduling decision of the other peers, and these decisions will influence the future scheduling of peer \( p \), and so on. In order to build an “omniscient” oracle, that knows future utilities in function of selections made by a peer, one must know the services costs for all peers, as well as their demands and their scheduling algorithms. Moreover, one must know the order in which the requests are made, and by which peers. This makes unattractive the design of an “oracle” function that can calculate the future utilities for each offered service.

A more suitable approach is to estimate the future utility based on historical data. One approach is the one that we use in this paper in order to assess the performance of the heuristics previously presented. We estimate the future utility according to a normal distribution. It is important to notice, however, that there is a chance of none of the services offered at turn \( t \) be consumed by other peers. This occurs if none of the peers that have \( s \) in their typical favor profile are consuming \( s \) at turn \( t \). The probability of this to happen depends on factors such as the amount of peers in the community, the number of different services demanded by peers (we call this value the size of typical favor \( |F_p| \)), and the amount of services that are being offered (this depends on the peers’ budget and the services’ costs).

Considering that the demand for all services is similar, the probability that a service \( s \) offered by a peer \( p \) is not consumed by a peer \( k \) is given by \((1 - |F_k|/n)\), and considering an universe of \( N \) peers, the probability that a service is not consumed by any peer is given by \( \prod_{i=1}^{N} (1 - |F_i|/n) \). Let \( \lambda_p \) be the average cost of services for peer \( p \), the average amount of services offered by this peer is given by \( (B_p/T\cdot\lambda_p) \). So, the probability of none of the services offered by peer \( p \) be consumed is given by the expression \( \text{ProbNoDon} \) below:

\[
\text{ProbNoDon} = \prod_{i=1}^{n} (1 - |F_i|/n) \left( \frac{B_p}{T\cdot\lambda_p} \right)
\]

Based on the above, the value of \( \lambda_{p,s,t} \) is given by the following bimodal distribution:

\[
\lambda_{p,s,t} = \begin{cases} 
0 & \text{with probability } \text{ProbNoDon} \\
X \sim N(\mu, \sigma) & \text{with probability } 1 - \text{ProbNoDon}
\end{cases}
\]

where \( \mu \) is the average and \( \sigma \) is the standard deviation of the services future utility.

C. Calculating the Performance of an Algorithm that Makes Random Selections

We develop a simple model to assess the performance of the trivial services selection algorithm that chooses the services to offer in a random way. To do so, we need to calculate the future utility that \( p \) attains, when it offers the service \( s \) at turn \( t \).

Considering the probability of none of the services offered by a peer be consumed by any peer, we can define an expression to determinate the future utility received from the grid by a peer that selects services randomly. Being \( \lambda_p \) the average future utility received by peer \( p \) from each unit of service provided, the expected future utility of \( p \) is given by:

\[
\text{TotalUtility} = (1 - \text{ProbNoDon}) \cdot \left( \frac{B_p}{T\cdot\lambda_p} \right) \cdot \lambda_p
\]

D. Performance Evaluation for a Scenario with Eager Consumers

We carried out experiments to measure the performance of the two first algorithms against the referential algorithm, which we implemented using a dynamic approach for the knapsack problem. Each experiment has submitted a set of knapsacks for each algorithm. The number of knapsacks was adjusted in each case, achieving a maximum error of 0.57% with a confidence level of 95%. In addition we calculate analytically the expected utility from peers that select services randomly using the expression \( \text{TotalUtility} \) described in the previous subsection.

The referential algorithm and the analytical results were used as benchmark, defining the “best” and the “worst” limits (we consider that an algorithm worse than a random selection is too naive to be considered).

In consonance with previous work [11], we have chosen to use a total of 300 different services, with costs following the distribution that were used in the experiments reported in that work: uniformly between 0.5 and 1. It is important to see that the total utility received from the grid is the sum of utilities delivered by each peer, and the sum of a sufficiently large number of independent random variables will be approximately normally distributed [25]. So, in order to define the future utility, we calculated the average results obtained by peers in the same experiments, finding an average utility \( \lambda\text{e_{p}} \) of 5.1 with a standard deviation of 2.5. So, we used this values to parameterize the normal distribution that estimates the future utility values.

Figure 1 shows the future utility received when using the Referential algorithm, the Cost-based and the Reciprocation-based heuristics, as well as the analytical model.
Since the costs are almost the same (see Figure 2), the final profit shows similar results, as we can see in Figure 3. It is important to see that the Cost-based heuristic obtained a poor performance, achieving results only marginally better than the analytical one. This occurs because the expected advantage obtained in selecting services by costs is to offer a larger set of services, increasing the probability to find a consumer. However, this advantage is small if the universe of services is large, because the proportional difference in number of offered services becomes negligible. On the other hand, the Reciprocation-based heuristic’s results are closer to the referential algorithm because this heuristic finds the more profitable services to be offered, although it cannot anticipate cases where the received profit differs from the expected value due to environment specificities.

The performance of the referential algorithm also allows us to assess the relative performance of the heuristics. Relative performance is defined as the comparative value between the heuristic performance and that of the referential algorithm. Figure 4 shows the relative performance of the Reciprocation-based and the Cost-based heuristics. When using lower budgets both heuristics achieve poor performances because the number of selected services is too small, decreasing the chances of finding a consumer. The greater is the budget, the greater is the number of possible selections, but the Reciprocation-based heuristics selects more profitable services than the Cost-based one and achieves better results. Nevertheless, relative performance of both heuristics increases when the budget increases. This occurs because with high budgets it is possible to select more services, decreasing the importance of choosing the correct services. In the limit, if the budget is large enough to offer all services, all algorithms would have the same performance.

E. The “invisible hand”

The results showed in figures 1, 2 and 3 considers an environment in which each unit of service offered by peers can be consumed, because the number of service units demanded by the grid is at least equal to the number of units of service offered. Due to this, the future utility (and also the profit) is directly proportional to the available budget.

However, when we consider environments where there are many peers offering multiple services under a limited demand, it is necessary to consider that there is a competition between peers. This kind of dispute limits the maximum profitability of peers, because it is necessary to consider the chance of “loosing to the competition”. This happens when a service is offered by many peers and some of them will donate no service, generating no utility (and a negative profit). This phenomenon limits the profit that can be achieved by peers, and is expected in competitive environments, according to the invisible hand of markets, as described by English economist Adam Smith.

In order to see the effects of competition, we must to define a fair competition environment. To do this, we consider that all peers in the community have the same budget. So, they can offer similar amounts of different services. Moreover, it is necessary to limit the amount of resources that can be consumed by the grid, creating a dispute of peers that are trying to donate services. So, in our experiments, the maximum amount of units that can be consumed for each service is equal to the capacity of a single peer. Therefore, the utility received by a peer must be divided with the other competing peers.

The greater is the average budget of other peers, the greater is the average number of different services offered by peers in the community. The greater is the number of different services offered by other peers, the greater is the probability of a peer to find competition when offering a specific service. So, we defined that the utility received by a service must be divided by
the expected number of peers that are also currently offering this service.

Figure 5. Average utility by turn under competition

Figure 6. Average cost by turn under competition

Figure 7. Average profit by turn under competition

Figure 5 shows the future utility achieved when considering environments with competing peers. Note that the average utilities are lower than the values showed in Figure 1, and have no direct relation between the budget and the received utility. This occurs because the heuristics choose to use all available budget to select services, making the cost grows up in a direct proportion with the budget. However, due to the competition, the utility has a logarithmic increase as we can see in Figure 5. As result, the final profit becomes negative when using higher budgets, as we can see in Figure 7.

The behavior of the Referential algorithm results can be considered “strange”, because it accepts to decrease the received utility for larger budgets, as shown in Figure 5. However, this can be understood when we look Figure 7. The analytically estimated profit, and that attained by the heuristic-based algorithms becomes negative, while the Referential algorithm stays in zero. This phenomenon is explained in Figure 6: knowing the future utility and cost values (and performing an exhaustive search in the solution space), only the Referential algorithm can find the most profitable values (and deciding for the selection of no item to put inside the knapsack, when all of them lead to negative values in function of competition.

V. RELATED WORK

Schahram et. al. [26] and Wang et. al. [27] have studied the services selection problem, but considering the composition of Web Services. These approaches address the problem of finding a workflow that links multiple peers together, and the balance of load on the network links [28], [29], [30].

The Network of Favors (NoF) [7] is a reciprocation mechanism that guide the exchange of resources in a P2P computational grid. Under contention, the preference is given to the peers who have donated more in the past. To control this, peers maintain a balance of past interactions with others. The balance expresses the amount of resources (called “favors”) each peer is in debt with others. Using the NoF, it is possible to marginalize free riders and give incentive for donations to collaborative peers. Mowbray et. al. [9], proposed an extended version of the NoF (ExtNoF) in order to deal with multiple services in a P2P grid environment. ExtNoF is a reciprocation-based mechanism to encourage donations where multiple services are shared explicitly, such as processing power, data transfer and storage. ExtNoF also addresses the problem of discouraging free-riding and enable peers to find others who have mutually profitable exchanges of services, even considering that peers value services differently. However, that work assumes that peers can offer all the possible services, which is not realistic in some cases where the provision of services is submitted to financial and hardware limitations.

A trust-based reputation mechanism is presented by Satsiou and Tassiulas to regulate the exchange of multiple services in P2P systems [31]. This work is similar to the one by Mowbray et al. [9], with the major difference being that, in the latter, the authors consider homogeneous peers with equal demands and capabilities. Both works show evidences that their reputation mechanisms leads to a dynamic formation of clusters among peers with mutually profitable interactions. However, none of them consider the fact of peers must select a subset of services to offer, and how the profitability can be improved by using clever selection methods.

There is an architecture, called Cluster on Demand (CoD) [32], that enables peers to donate their idle machines, defining how many machines must be donated to each other. The set of machines received by peers is perceived as a “virtual cluster”, which is a pool of configured machines that are
donated by different peers, allowing the receiver to use it as a real cluster. Moreover, the donating peer is able to configure the machines with multiple services, according to the receiver needs. However, the scheduling policy must be defined previously by peers, and services are configured in the machines after that. So, differently from the scheme studied in our work, the CoD proposal neither considers an automated configuration environment, nor takes into account a limited budget to offer services. Moreover, heuristics algorithms can be used with CoD, or any similar architecture, in order to improve peer’s profitability, and their performance can be evaluated using the methodology described in this work.

Peers need to take management decisions in order to offer services to the community aiming to improve their profitability. Although our previous work has shown the importance of appropriate services selection [10] and heuristics to select services [11], it is necessary to determine a methodological way to compare services selection and heuristics performance. To the best of our knowledge, there is no work that has addressed a methodological way to evaluate the performance of different algorithms used to select services in a multi-service P2P grid.

VI. CONCLUSIONS AND FUTURE WORK

To evaluate the performance of algorithms is a crucial step when considering environments with high computational complexity because these algorithms usually need to use heuristic approaches. Methodologies are normally based on referential algorithms that are used as an upper bound when evaluating heuristics performance. However, in the services selection problem, the evaluation by using referential algorithms is quite difficult to be performed due the complexity of the system.

To avoid this obstacle, we proposed a methodology using a simpler version of the problem that is an instance of the well-known knapsack problem. Using this methodology a referential algorithm, that executes a brute-force search in the solutions space, can be implemented and used as benchmark to evaluate heuristic methods for the services selection problem.

The mapping of the services selection problem onto the knapsack problem is made possible by introducing additional knowledge about the future utility that a peer will obtain, when a given service is provided. On the other hand, methods under evaluation must perform the selection of items without knowledge about future utility. This information must be hidden because, in practice, peers do not know the future utility yield by the provision of services.

In our experiments we evaluated two heuristics using this methodology, considering environments with and without competition. When we consider a competitive environment, both heuristics achieved a lower performance when compared to the results achieved in the environment without competition. This occurs because the cost increases in a direct proportion to the available budget, while the utility is limited by the competition. This may make the final profit negative in some cases. As a future work we intend to develop heuristics that search for the best budget to be used. Moreover, considering that peers made a sequence of selections along the time, we intend to investigate the use of machine learning algorithms in order to design heuristic methods that allow peers to progressively improve their profit.

In this work we varied the peers’ budget in order to understand how the number of offered services impacts on heuristics’ performance. But there are many other environmental factors that may have an effect on the peer’s profitability: the typical favor size, the amount of peers in the community, the number of different services in the environment and the grid contention. We intend to define experiments considering all of these aspects in order to investigate the use of algorithmic methods to improve the profitability under different scenarios.

Another future work is to compare the performance of heuristics when using this methodology with results found in real P2P grid environments by performing experiments using simulations and prototypes of real P2P grids.

REFERENCES


