Evaluation of Operation Research Approaches to Solve Allocation Problems in Decentralized Networks

Robert Schirmer
Betreuer: Dipl.-Inf. Matthias Wachs
Seminar Innovative Internettechnologien und Mobilkommunikation SS 2013
Lehrstuhl Netzarchitekturen und Netzdienste
Fakultät für Informatik, Technische Universität München
Email: schirmro@in.tum.de

ABSTRACT
Availability and quality of service are important aspects of the Internet, as it relies on many independent components being able to communicate. This is why the stability of the Internet and the ease of acquiring informations through it can be jeopardized by even small incidents, accidental or not. As a result, a lot of work is done in order to improve the availability, robustness and general quality of networks. A very promising area is to make networks more flexible by supporting different communication mechanisms (such as the use of protocols that are cryptographically more resistant), thus making the network more tolerant to censorship and hardware failures. The ability to use many different protocols brings with it an allocation problem: how much should be sent via which protocol in order to achieve maximum satisfaction? In this paper, we will look at heuristic techniques to answer that question.

Keywords
Operations Research, Decentralized Networks, Resource Allocation Problem, Heuristics, Metaheuristics, Optimization

1. INTRODUCTION
Nowadays, computer networks such as the Internet are the core of modern communication and are essential to many aspects of everyday life. There exist a lot of factors that explain our societies dependence on them, but two important ones are certainly its widespread availability (how easy of access and robust it is) and its performance (how quickly data is transferred compared to other mediums, be it latency or bandwidth wise). The robustness, which is a factor for the availability of networks is of particular interest to us, as the communication can quickly come to a stop or be severely hindered by incidents: physical (for example the sectioning of a fibre glass cable), accidental (such as a Danish policeman confusing DNS censoring tables and locking the access to about 8000 websites for many users [19]), political (as seen in the example of the Egyptian government blocking access to different social networking services like Twitter or Facebook [20]) or economical (Internet Service Providers (ISP) throttling certain applications or protocols in order to reduce unwanted traffic). Increasingly, different actors are trying to have an influence over the information flow on the Internet to suit their own needs, such as the ISPs or governments in the examples earlier, but application developers are adapting to this situation and try to circumvent these restrictions by using various techniques like encryption, dynamic port assignment and port hopping.

In this paper, we will have a closer look at a peer-to-peer network called GNUnet, it is a framework that proposes peer-to-peer functionality for any implementing applications. GNUnet is focused on network security and on improving connectivity. Network security is important, as it is focused on avoiding disruptions, outside interferences or guaranteeing the anonymity of peers. Connectivity has two sides in the context of GNUnet, one in restricted environments (like inside a firewall or using ISP filtered traffic) where the focus is on avoiding the restrictions and enabling the network to be more efficient, and the other in infrastructure-less (adhoc) networks, where there are different kinds of restrictions, such as the relative instability of the transport medium. In order to master these restrictions, GNUnet supports multiple transport mechanisms, as this offers the possibility of using the mechanism that proposes the best possible performance, connectivity and censorship-resistance for each pair of peers. This gives us very good robustness, as the flexibility makes the network resistant to the blocking of ports and other hindrances: if for example TCP on port 2086 is blocked by the ISP, the program can just switch to a HTTPS connection. GNUnet has several additional properties: the user can limit the used bandwidth and, as it is a framework, it is used for a wide array of applications that have their own QoS requirements that should be respected. A file-sharing service would for example privilege higher bandwidth while a VoIP program would prefer a lower latency. Between these two extreme examples lie many applications with varying service requirements, such as electronic commerce, video conferencing and online gaming. All these applications require the allocation of enough bandwidth for satisfactory performance. Because of the limited network resources and the different abilities of the supported protocols, it is obvious that some kind of conflict appears when one tries to decide who gets what share of the bandwidth and via which protocol. To solve the conflict of applications with different QoS values that compete for the limited bandwidth in the context of multiple transport mechanisms with different performances is the objective of the Resource Allocation Problem (RAP) in a decentralized network, which we will look at in this paper.

It is important to differentiate the routing problem in networks and the RAP: the routing problem is situated on the Network Layer of the OSI reference model while the RAP is located on the Transport Layer. This means that the routing problem is focused on routers, bridges or switches and is trying to find optimal ways from a user to another or how to avoid congestions. The RAP supposes that this problem is already solved and uses its results in order to decide which user should use how much bandwidth and protocol in order to attain maximum QoS.

In optimization, there are many ways to find the best solution to a given problem, but they can broadly be put into two categories: exact techniques and heuristic techniques. Exact approaches are used when the search space is rather small or the problem has a special structure so that not every solution needs to be tested. Heuristic procedures are used when the search space is very big and it would take too long to come up with an exact solution; this is why in most real-life problems heuristic methods are privileged, because the trade-off between solution quality and computation time is critical. Operations Research (OR) is a vast field of study that is focused on optimization, simulation and representation of many real life problems; it is a field that has existed for a very long time and has been known to produce very good techniques to solve many large instances of NP-problems to near optimality. This is the reason this paper is focused on OR techniques, as they seem very promising for the problem at-hand.

So as not to ”reinvent the wheel” each time one tries to solve a problem, it is practical to map it to known problems that are treated in the literature. For example, if we wanted to solve a Vehicle Routing Problem, which is a generalization of the well-known Travelling Salesman Problem (TSP) where there is more than one salesman, we could simply decide to form as many ”clusters” of cities as we have vehicles, and solve the resulting TSPs using techniques that are well known-and-used.

The rest of the paper is organized as follows: Section 2 describes the RAP formulation and its relationship to Operations Research, Section 3 describes the principles of heuristic optimization and their applicability to the RAP followed by the conclusion in Section 4.

2. THE RESOURCE ALLOCATION PROBLEM IN DECENTRALIZED NETWORKS

In our problem, we will study a peer-to-peer network such as figure 1, where a certain amount of computers are connected to each other directly or indirectly in different kinds of networks, such as Wide Area Networks (WAN) or Local Area Networks (LAN) etc... We will abstract from the routing proceedings of the network and consider its topology given, which is why we will only take them into account in form of such measurable metrics like latency, maximum bandwidth and the amount of hops necessary to reach a peer. In the context of our problem, the network is still considered dynamic, which means that the attributes change over time, or peers could come and leave.

While one could imagine that solving the general problem for the entire network by exchanging information between the peers would yield better results, it would be unwise to do that for the following reason: In a peer-to-peer network, other peers are not always to be trusted, as they could be feeding false information into the network in an attempt to render the global best solution useless. This issue is a variant of the Byzantine Generals Problem [13], where one has to try to calculate an optimum using conflicting informations. To solve this problem is computationally expensive which is why, in our paper, every peer will try to solve his local optimal solution without exchanging information with his neighbours.

2.1 The Resource Allocation Problem

It is useful to examine how peers communicate with each other in order to understand the problem further: We will call a transport mechanism to designate a protocol or method to transmit data between different peers with specific properties or constraints. An address specifies precisely how to reach a particular peer and includes the transport mechanism (for example, TCP) and the specific network address (for example, 1.2.3.4:5555). A peer may be reachable under many addresses and even if two addresses use the same transport mechanism, they are considered separate in our problem, as the performance might be different. For example, TCP may be used to communicate with a peer via an IPv6 link-local address, an IPv6 global unicast address or an IPv4 address. Our algorithm must then choose between three different addresses for this peer.

Moreover, every peer-to-peer application that runs on the members of the network has certain preferences that need to be respected in order to give the user a maximum satisfaction. For example, a peer-to-peer video chat application, such as Skype, would need small latency and enough bandwidth to allow video communication.

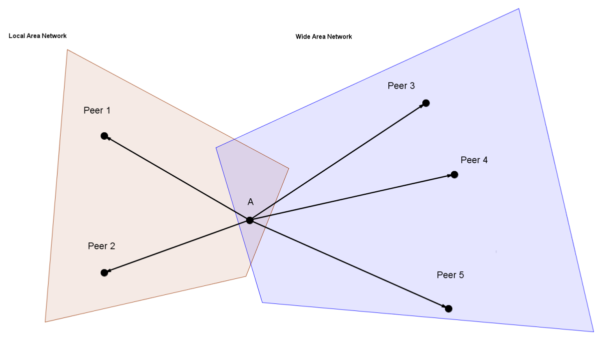

Figure 1: The RAP in a decentralized network. *We are looking at the network from the perspective of A, who is in contact with 5 peers via WAN or LAN. Each one of the peers can be reached via different transport mechanisms.*

The next step in the explanation of the problem is the objective function, what we are trying to achieve when in front of this problem. In certain contexts, the words ”fitness function” are used instead of ”objective function”, they both refer to a quantification of solution qualities that are used to compare two different solutions. The idea here is that we try to maximize the used bandwidth in such a way that maximizes the quality of service to the user, hence that suits the application preferences best. Obviously, none of the outgoing connections via a mechanism can exceed that mechanisms capacity. The output of the optimization process is a set of mechanisms and the assigned bandwidth per mechanism in order to maximize the objective function.

With $p \in P$ a peer from the set of peers, $m \in M$ the metric from the set of metrics (where a larger value means a better performance) and $t \in T_p$ a transport mechanism from the set of transport mechanisms for a peer p, we have: The performance preference for each peer is expressed using values $f_{p,m}$. There is no unit, the value is completely application specific and only the ratio between the various $f_{p,m}$ matters. $q_{t,m}$ specifies the current quality rating for the metric m using the mechanism t. Our objective is to maximize $\sum_{m \in M} \sum_{p \in P} \sum_{t \in T_p} f_{p,m} \cdot q_{t,m}$. There are also some more aspects to the allocation problem that are not in the focus of this paper, like the fact that peer-to-peer networks tend to function better when many peers are in contact with each other, which is why that could be present somewhere in the objective function.

In short, our problem includes:

- A focus on the RAP (how much to communicate via which transport mechanism), we will not talk about the routing problem (how to reach a user), which is why we will look at the network from a user-to-user perspective.
- A peer-to-peer network where users communicate data with each other.
- A problem from the perspective of a peer, as in a peer-to-peer network nobody manages the data flow.
- Potentially many transport mechanisms to communicate with a peer.
- Preferences for each application used by a peer, which have a big importance for the objective function.

2.2 Theoretical Background

At first, we will prove that our problem is NP-hard, which means that for every peer that is added, the additional computation time required to find the solution to the bigger problem is non-polynomially greater than for the smaller problem. At first, we will examine the Knapsack Problem, a well known problem in theoretical computer science. According to [4], we are given a set $S = a_1, a_2, \ldots, a_{n-1}, a_n$ a collection of objects, $s_1, s_2, \ldots, s_{n-1}, s_n$ their sizes and $p_1, p_2, \ldots, p_{n-1}, p_n$ their profits and a knapsack of capacity B. The goal is to find a subset of objects whose total size is bounded by B and whose profit is maximized. This problem is known to be NP-hard. It is obvious that our RAP contains many instances of this problem: for each network type (WAN, LAN etc...), we have a collection of objects (namely all the transport mechanisms available for each peer at every bandwidth possible), their sizes in comparison with the limited resource (the bandwidth) and their profits (namely the profit function, which is a function of the bandwidth and other aspects of the network). In this way, we have shown that our RAP is NP-hard.

The Resource Allocation Problem in a decentralized network (RAP) is a special case of a Multi Commodity-Flow problem (MFP), which is a standard problem in the literature. The Resource Allocation Problem (not in a decentralized network!) is an optimization task where one wants to map a limited amount of resource to a number of tasks for optimizing their objectives. The complexity of this problem is obviously related to the functions that calculate the amount of use generated by the resources for each task. In practice, commodities may represent messages in telecommunications, vehicles in transportation or product goods in logistics. The RAP has a variety of applications, including product allocation [21], resource distribution [3], project budgeting [9], software testing [22], processor allocation [2] and health care allocation [12], just to name a few. For a deeper overview of the subject, the reader is referred to [18].

Multi commodity-flow problems are characterized by a set of commodities to be routed through a network at a minimum cost, as can be seen in figure 2. The MFP is defined on an undirected graph $G = (V,E)$ with an assignment of non-negative capacities to the edges, $c : E \to \mathbb{R}_{\geq 0}$. A MFP instance on G is a set of ordered pairs of vertices $(s_1,t_1), (s_2,t_2), \ldots, (s_k,t_k)$ (not necessarily disjoint). Each pair (s_i,t_i) represents a commodity with source s_i and target t_i. The objective is to maximize the amount of flow travelling from the sources to the corresponding destinations, subject to the capacity constraints [17]. There are different flavours of the problem, like the fractional MFP where for each commodity (s_i,t_i) a non-negative demand d_i is specified. The objective is to maximize the fraction of the demand that can be shipped simultaneously for all commodities, as seen in figures 3 and 4. When, in the context of a MFP, the capacity of the edges is set to be infinite and the objective is not to maximize the amount of flow, but the use that is generated from the flow, we obtain a Resource Allocation Problem again. As long as the function that maps the flow to the utility of the flow is linear, then the problem is of the same complexity class as the MFP.

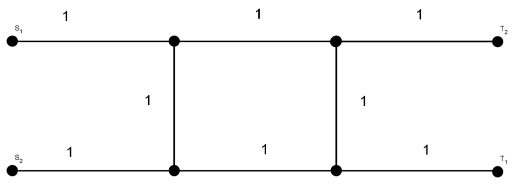

Figure 2: A MFP

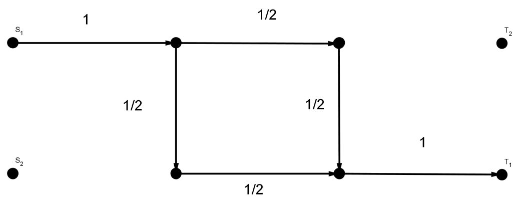

Figure 3: The solution to the MFP for commodity 1

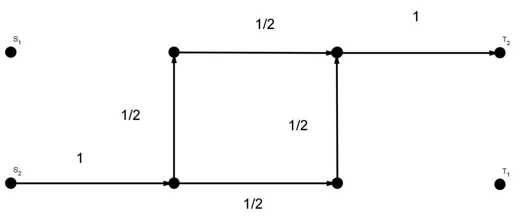

Figure 4: The solution to the MFP for commodity 2

As we can see, the RAP in decentralized networks which we are treating right now is a special case of the MFP. We have the case of having only one source and many targets, of having a network that has some very specific capacity constraints (such as the fact that only one transport protocol can be used for each peer) and of having a cost function that is relative to the bandwidth reaching the peers and the quality metrics. It is very difficult to model this as a MFP, but it is possible by solving a new MFP for each of the subproblems, and comparing the values of the objective function. To illustrate that, we have in figure 5 the structure of the aforementioned peer-to-peer network as a MFP, and in figures 6 and 7 how to solve it repeatedly in order to figure out a solution for our RAP. This means we have *amount of transport mechanisms*$^{number\ of\ peers}$ MFPs to solve. Even though the amount of computation needed to solve this is enormous for bigger instances, we can see that the problem structure is the same for the subproblems of the RAP and the MFP.

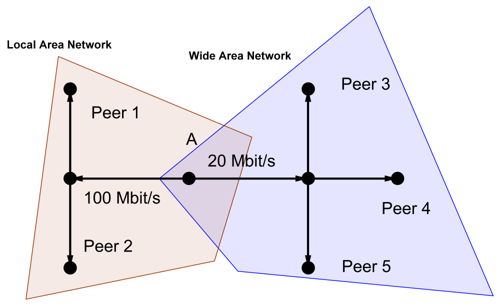

Figure 5: The RAP as a MFP

The optimization process can take place in several different ways, such as Integer-Linear Programming (ILP), which in the literature has been known to produce the best results for the MFP. In ILP, the exact solution of the problem is found out. This has its advantages, as using the best solution is the one that promises the best performance and quality of service for the peers. But finding the best solution is often a relatively long process; As we are in the context of a dynamic network, where application requirements and network states change very often, all the while the package transmission time is of the order of a few hundreds milliseconds, it does seem very interesting to shorten the optimizing in order to lengthen the actual transmission of data.

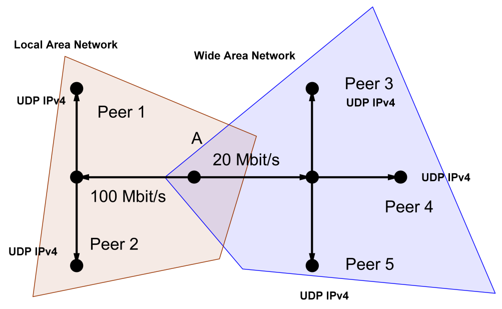

Figure 6: The first decomposition of the RAP as a MFP
We are supposing the LAN has 100 Mbit/s capacity, and the WAN 20 Mbit/s capacity

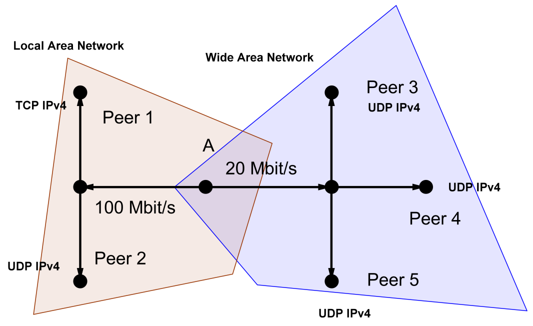

Figure 7: The second decomposition of the RAP as a MFP

This is where Operations Research (OR) comes in handy, as it is a discipline that is bent on solving complex combinatorial problems in a limited amount of time. Often in OR, the optimal solution is extremely difficult to calculate because of the size of the search space (like is the case for NP-problems, such as ours). But, as OR has its roots in the practical, a "good" solution that is calculated quickly is usually completely satisfying. A solution that is computed in half as long as the optimal solution and that maximizes the objective function to 80 % of the optimal solution can be extremely interesting in our case. This is why, in this paper, we will look at four different metaheuristics from OR and see how they could be applied to suit our purposes and compare them.

3. METAHEURISTICS

Before speaking about metaheuristics, we must define a few important aspects that will help us in the following explanations. At first, it is worth noting that we are in front of a combinatorial optimization problem, as we must find an optimal resource allocation from a finite set of possible allocations. But it is capital to devise a way to "navigate" from a solution to another one. This is called the neighbourhood structure, as we are defining what solutions are considered neighbours and by which operator we can go from one solution to the next. We propose an operator where we differentiate solutions by the peers that are communicated with, the used transport mechanism and the amount of bandwidth assigned through this mechanism. The operator to navigate through solutions would be x kB/second more or less on any transport mechanism to any peer (with $x \in \mathbb{N}^*$). Obviously, this operator can bring us infeasible solutions (where many transport mechanisms are used to communicate with a peer or other constraints are not respected), which is why we must always keep an eye out for them before applying the operator). We can also use a different step, resulting in a changed search space (for example if we use a step of 5x kB/second, the search space would be 5 times smaller). This means our neighbourhood can be defined as "the set of all solutions that are reachable by changing the bandwidth with any transport mechanism by x kB/second". This neighbourhood structure obviously has its flaws, as it is impossible to switch transport mechanisms directly: one must first follow the neighbourhoods of solutions to a point where there are no transport mechanisms for a peer, and only afterwards one can start analysing another one.

To get back to metaheuristics, they usually take a solution as an input, which in our case, is the peers we are communicating with, their transport mechanism and their assigned bandwidth. The starting solution can be far from optimal, it is just needed to have a point in the search space where we can start improving.

3.1 Improvement heuristics

Improvement (Hill Climbing) heuristics are heuristics that take a solution as input, modify this solution by performing a sequence of operations on the solution and produce a new, hopefully improved solution. At all times the heuristic has a current solution and it modifies this solution by evaluating the effect of changing the solution in a systematic way. If one of the changes leads to an improved solution, then the current solution is replaced by the new improved solution and the process is repeated. The input solution might come from any source, but it appears that the better the starting solution, the quicker the optimum will be reached. As we are still in the domain of classical heuristics, we usually only consider the neighbourhood of our current solution.

The problem here is that while we are only considering direct neighbourhoods, the search will probably get stuck within a local optimum, as can be seen in figure 8. This means that there are no more better solutions within the direct neighbourhood, while there might still be a better solution somewhere out there if one were to consider a wider neighbourhood (using a different kB/second step). To escape these local optima, metaheuristics can search the solution space in a broader way and are, on the long run, almost always better than classic heuristics. We can discern between metaheuristics that have a population of solutions that are being iterated upon and metaheuristics that iterate upon a single solution. Simulated Annealing and Tabu Search are in the latter category while Genetic Algorithms and Ant Colony Optimization belong in the first one.

3.2 Tabu Search

Tabu search (TS) is another popular search technique proposed by Glover in 1977 [8]. Since then, it has been widely used for solving CO problems. Its name is derived from the

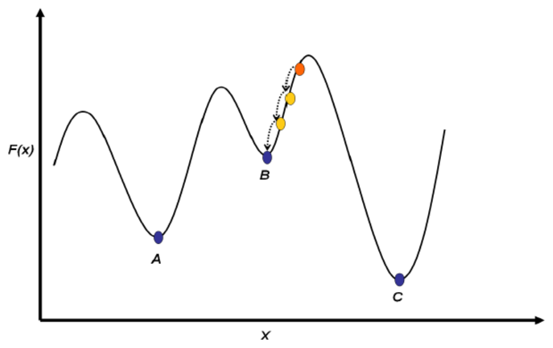

Figure 8: Graphical representation of a Hill Climbing Algorithm - downhill climbing from an initial solution to a local optimum
X is the solution space and F(x) is the value of the objective function for that particular solution.

word "taboo" meaning forbidden or restricted. TS, like SA, allows for exploring the search space "intelligently" in an attempt to escape the trap of local optima. There are many variants of TS that all have their specificities and operators. Nevertheless, there are two main attributes that all variants of TS have.

1. They all allow the current solution to deteriorate, as opposed to normal Hill Climbing algorithms,
2. Every TS uses a short term memory called a tabu list, in which moves that have been recently visited during the search are recorded. Moves in the tabu list are considered prohibited by the search and cannot be visited again for a certain number of iterations. The idea is to avoid the problem of cycling, meaning that the search may be trapped within the boundaries of a certain neighbourhood region, oscillating among solutions that have been previously visited, as illustrated in the next figure. By prohibiting recently visited moves, the algorithm is forced to explore new areas of the search space in an attempt to escape local optima.

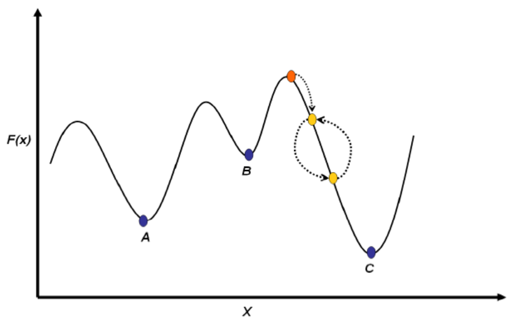

Figure 9: The Problem of Cycling

It is also sometimes fruitful in TS to make use of an intensification and/or a diversification mechanism. Intensification tries to enhance the search around good solutions, while diversification tries to force the algorithm to explore new search areas, in order to escape local optima. For example, intensification can be performed by encouraging solutions that have some common features with the current solution. On the other hand, diversification may be enforced by applying a penalty in the objective function, at some stage of the search, to solutions that are close to the present one [5]. Some variations of TS also exist in the literature, for example Probabilistic TS assigns a probability to neighbourhood moves, such that some attractive moves that lower the solution cost are given a higher probability, while moves that result in a repetition of some previous state are given a lower probability [7]. Also, a Reactive TS was proposed by Battiti and Tecchiolli [15] in which the size of the tabu list is adapted dynamically during the search, according to the frequency of repeated moves. Another variant by Cordeau [11] proposes a possibility of looking at infeasible solutions in order to navigate the search space more intelligently.

The latter variant is the most interesting one for our RAP, as the neighbourhood structure has a lot of infeasible solutions in it: navigating them intelligently would be a huge benefit for the runtime of a metaheuristic. TS has the advantage of having a reasonable computation time and to offer very high quality solutions. Many Combinatorial Optimization problems in the literature have been solved to near-optimality with TS and in very interesting runtime. The problem is that TS often lacks robustness: the efficiency depends on several factors that can alter parameters which are problem specific. TS is nevertheless apt to handle dynamic problems, as it is relatively quick and we can alter solutions "on the go" and keep on optimizing each time the problem is updated.

3.3 Simulated Annealing

The theoretical foundation of Simulated Annealing (SA) was led by Kirkpatrick et al. in 1983 [16]. SA is a well-known metaheuristic search method that has been used successfully in solving many combinatorial optimization problems. It is a stochastic relaxation technique that has its origin in statistical mechanics. The Simulated Annealing methodology is analogous to the annealing processing of solids. In order to avoid the less stable states produced by quenching, metals are often cooled very slowly which allows them time to order themselves into stable and structurally strong low energy configurations - This process is called annealing - This analogy can be used in combinatorial optimizations with the states of the solids corresponding to the feasible solution, the energy at each state to the improvement in objective function and the minimum energy being the optimal solution. SA involves a process in which the temperature is gradually reduced during the simulation. Unlike Hill Climbing, SA is a global optimization heuristic based on probability therefore is able to overcome local optima. However, although it yields excellent solutions, it is very slow compared to a simple hill climbing procedure. When solving a Resource Allocation Problem using SA, we start with a certain feasible solution to the problem. We then try to optimize this solution using a method analogous to the annealing of solids, as can be seen in figure 10: A neighbour of this solution is generated using an appropriate method, and the value of the objective function of the new solution is calculated. If the new solution is better than the current solution in terms of increasing the use of network, the new solution is accepted. If the new solution is not better than the current solution, though, then the new solution is only accepted with a certain probability. The SA procedure is less likely to get stuck in a local optimum, compared to a classical Hill Climbing heuristic, since bad moves still have a chance of being accepted. The annealing temperature is first chosen to be high so that the probability of acceptance will also be high, and almost all new solutions are accepted. The temperature is then gradually reduced so that the probability of acceptance of low quality solutions will be very small and the algorithm works more like hill climbing, i.e., high temperatures allow a better exploration of the search space, while lower temperatures allow a fine tuning of a good solution. The process is repeated until the temperature approaches zero or no further improvement can be achieved. This is analogous to the atoms of the solid reaching a crystallized state.

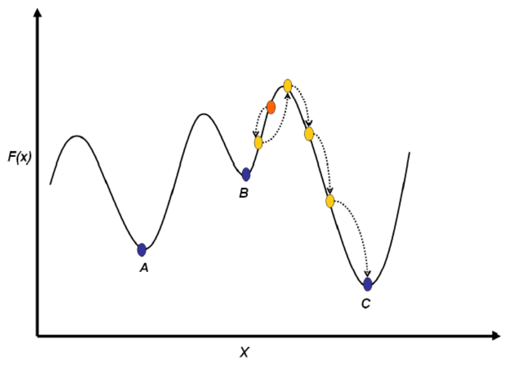

Figure 10: Simulated Annealing - occasional uphill moves and escaping local optima.

It appears obvious that the efficiency of Simulated Annealing relies on many factors, such as:

- What is the initial probability of accepting a bad solution given a certain time? (initial temperature)
- How many iterations should be carried out at each temperature? (or new value of probability)
- What is the objective function? As it's definition may lead to the rejection of promising leads or acceptance of solutions that are already infeasible, whatever happens.
- The topology of the neighbourhood structure is also critical to the performance of the SA algorithm. In general, a smooth topology with shallow local optima is favoured over a bumpy topology with many deep local minima.

The basic advantages of SA are the following:

1. It is very easy to implement, since it just requires a method for generating a move in the neighbourhood of the current solution, and an appropriate annealing schedule.
2. High quality solutions can be obtained using SA, if a good neighbourhood structure and a good annealing schedule have been chosen.

With the solving of the RAP in mind, SA doesn't seem like a very good solution, as the neighbourhood structure is very bumpy due to the difficulties of finding a feasible solution in the neighbourhood. This forces the neighbourhood function to not be able to "switch" transport mechanisms, resulting in a very hilly solution space. Another problem of SA is that due to its nature, it only considers one solution at a time, which for a search space as huge as the one we are considering means a pretty slow computation time. The dynamic nature of our problem makes SA badly suited and well suited at the same time, bad as it would be a waste to spend so much time for a near optimal solution. Useful when the setting changes, we can just go on iterating from the best solution found before the change. This way, we can hopefully profit from the work done before by just "changing" the actual solution to fit the current new problem. (deleting a peer if he has stopped contributing to the network and go on iterating for example).

3.4 Ant Colony Optimization

Ant Colony Optimization (ACO) is a metaheuristic technique that is inspired by the behaviour of real ants. The general reasoning behind the algorithm is that a colony of ants can always find the shortest path between the nest and the food source, despite the fact that every individual ant is blind, how do they do this?

Its principles were established by Dorigo et al. in 1991 [14]. Real ants cooperate to find food resources by laying a trail of a chemical substance called pheromone along the path from the nest to the food source. Depending on the amount of pheromone available on a path, new ants are encouraged, with a high probability, to follow the same path, resulting in even more pheromone being placed on this path, as seen in figure 11. Shorter routes to food sources have higher amounts of pheromone. Thus, over time, the majority of ants are directed to use the shortest path. This type of indirect communication is called stigmergy, in which the concept of positive feedback is exploited to find the best possible path, based on the experience of previous ants.

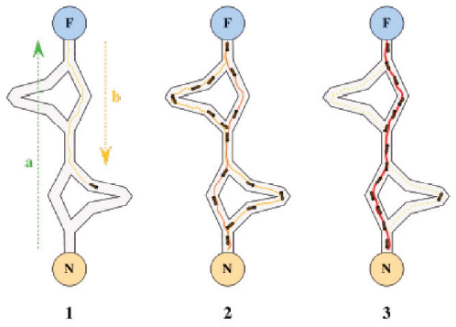

Figure 11: Colony converging on the shortest path - [1]

In our case, artificial ants (software agents) cooperate to find the best path by constructing solutions consisting of transport mechanisms and the assigned bandwidth step-by-step. Each step adds a selected transport mechanism (and it's bandwidth) to the partial solution in a stochastic manner, but biased by the amount of pheromone available on the transport mechanisms. One can also have the probabilities be influenced by classical heuristics that were run on the problem beforehand. To take into account the work of the other ants, problem-specific information is placed in a common memory, which plays the role of the pheromone trail. The information in the memory, i.e., the pheromone trail, is updated by ants at each iteration, allowing ants to cooperate in finding good problem solutions. Pheromone values also diminish over time, similar to the evaporation of real pheromone. Thus, solutions of bad quality are eliminated from further consideration as the search progresses.

To solve a Resource Allocation Problem using Ant Colony Optimization, we must first send many artificial ants free in the wilderness of the solution space. To do this, we could use an ACO structure that is a combination of figures 12 and 13, in order to achieve a 2-dimensional structure to optimize for the bandwidth and the transport mechanism to use. We consider using the bandwidth mechanism of figure 13, where the bandwidth is chosen randomly from the given interval, because this way we can search the space better (it seems obvious that the problem size increases exponentially when we have more segmentations of the bandwidth). We should start off by creating many random solutions sequentially and setting a stronger pheromone path on solutions that, by some "accident", are better. This way, in the next iterations of the ACO, we can use stochastically enhance the search around the paths of solutions that have proven interesting. In the long run, the whole search space should be covered and the result should be a good solution.

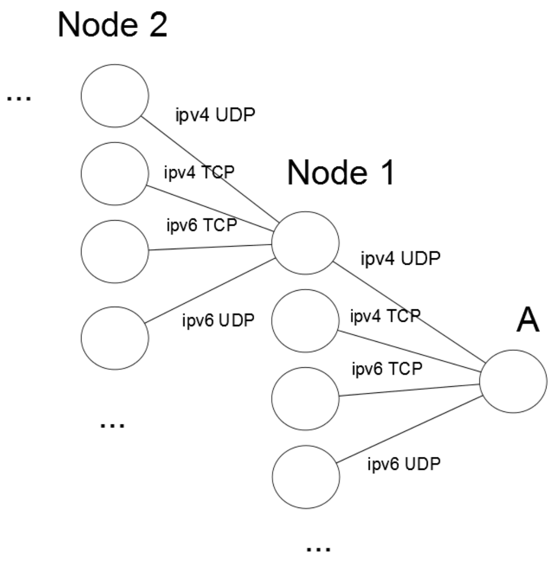

Figure 12: Our ACO in order to find which transport mechanism to use.

When originally presented, ACO was inferior to state-of-the-art heuristics and metaheuristics used to solve its first problem, the TSP. ACO is, on the other hand, very flexible and applicable on a large array of problems. For further information about the various type of Ant systems and their applications, the reader is referred to the book by Dorigo and Stuetzle [6].

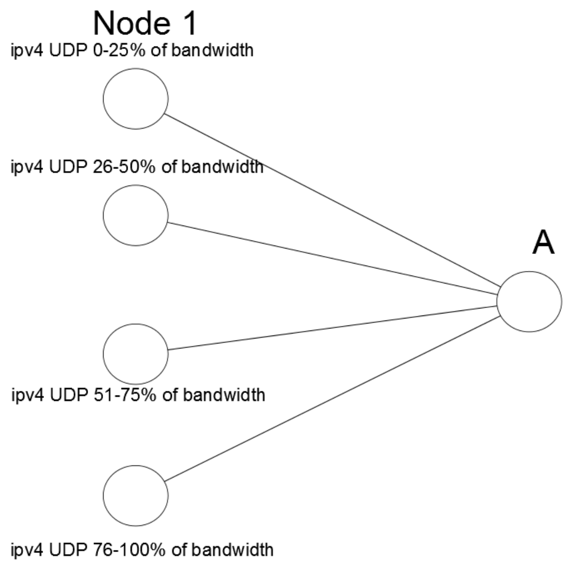
Node 1
ipv4 UDP 0-25% of bandwidth
ipv4 UDP 26-50% of bandwidth
A
ipv4 UDP 51-75% of bandwidth
ipv4 UDP 76-100% of bandwidth

Figure 13: Our ACO in order to allocate the bandwidth for the medium (WAN, LAN etc...) to each transport mechanism to use.

ACO has several attractive features for the solving of the RAP:

1. Inherent parallelism: we easily find ways to execute it in parallel, which is good for the runtime.
2. Positive Feedback accounts for rapid discovery of good solutions.
3. As it is swarm-based, we can easily analyse a broader part of the search space.
4. It suits the dynamic part of the problem well, as in case of a change, we just need to update the pheromone-matrix (by adding or deleting paths when a peer is added/deleted) and we can get the algorithm running again using the pheromone matrix, hence keeping all the work we did up to now.

3.5 Genetic Algorithms

The idea of simulation of biological evolution and the natural selection of organisms dates back to the 1950's. Nevertheless, the theoretical foundation of GAs were established by John Holland in 1975 [10], after which GAs became popular as an intelligent optimization technique that may be adopted for solving many difficult problems.

The idea behind GA is to simulate the processes of biological evolution, natural selection and survival of the fittest in living organisms. In nature, individuals compete for the resources of the environment, and they also compete in selecting mates for reproduction. Individuals who are better or fitter in terms of their genetic traits survive to breed and produce offspring. Their offspring carry their parents basic genetic material, which leads to their survival and breeding. Over many generations, this favourable genetic material propagates to an increasing number of individuals. The combination of good characteristics from different ancestors can sometimes produce super fit offspring who out-perform their parents. In this way, species evolve to become better suited to their environment.

GAs operate in exactly the same manner. They work on a population of individuals representing possible solutions to a given problem. In traditional GAs, each individual is usually represented by a string of bits analogous to chromosomes and genes, i.e., the parameters of the problem are the genes that are joined together in a solution chromosome. A fitness value is assigned to each individual in order to judge its ability to survive and breed. The highly fit individuals are given a chance to breed by being selected for reproduction. Thus, the selection process usually favours the more fit individuals. Good individuals may be selected several times in one iteration, while poor ones may not be selected at all. By selecting the most fit individuals, favourable characteristics spread throughout the population over several generations, and the most promising areas of the search space are explored. Finally, the population should converge to an optimal or near optimal solution. Convergence means that the population evolves toward increasing uniformity, and the average fitness of the population will be very close to the highest fitness.

Chromosome representation as a bit string is not suitable for many problems types, though. Which is why for the RAP, we will use the chromosomes as described in figure 14. There are as many of these genes as there are peers in our network. We need to use this representation in order to avoid the problem of generating offspring that violate constraints, such as having more than one transport mechanism for a peer. In order to execute the crossover operator, we would just need to exchange transport mechanisms and allocated bandwidths from the same peer to ensure conformity to the constraints. To execute the mutation generator, which is used to get some diversity into the algorithm we can just change the transport mechanism altogether and/or the allocated bandwidth.

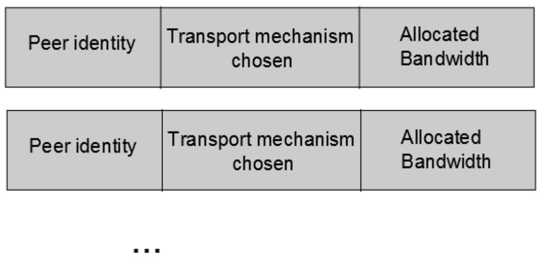

Figure 14: How a chromosome would look like for the GA solving the RAP

In respect to the RAP, GAs represent an intelligent swarm based search method which allocates higher resources to the promising areas of the search space. GAs have the advantage of being easily hybridized and to combine the genetic paradigm with some kind of local search to make even better solutions. GAs have been known to produce very good results on large instances very quickly, which is also of great interest to us. The dynamic context is also something that profits to GAs, as the optimization process can go on unhindered after changes in the setting have been applied in the fitness function.

4. CONCLUSION

In this paper, we have first introduced a problem that isn't present in the literature (namely the Resource Allocation Problem in decentralized networks), and then seen how it relates to other standard problems. After having shown that the problem is NP-hard, we have shown how OR techniques can be used in order to solve the problem non optimally while keeping an eye on the runtime. It is important to note that due to the current state of affairs where we have a linear objective function (namely more bandwidth makes the quality of service linearly better for a given transport mechanism), the structure of the problem is such that dynamic programming, more specifically Integer Linear Programming can take advantage of. But if in the future a more realistic approach is taken using more individualized functions, thus representing the world better, then the search space loses the structure that makes dynamic programming the optimal choice. Solving the new problem would certainly rely on metaheuristics, as the resulting search space would be too complex to master. The most promising approach is hybridizing a swarm based metaheuristic like Genetic Algorithms with a local search one like Tabu Search, as the swarm based heuristic can guide the problem solving in a more general way, while the local search one can help navigate the extremely intricate search space in a more local way.

5. REFERENCES

[1] Shekhawat A., Pratik P. and Dinesh B. Ant colony optimization algorithms: Introduction and beyond. -, 2009.

[2] M. Krishnamoorthy, A. Ernst and H. Hiang. Mathematical Programming Approaches for Solving Task Allocation Problems. 2001.

[3] P. Zipkin and A. Federgrün. Solution Techniques for Some Allocation Problems. *Mathematical Programming*, 1983.

[4] L. Cowen. The Knapsack Problem. University Lecture, 2009.

[5] A. Hertz and D. de Werra. Tabu Search Techniques: A Tutorial and an Application to Neural Networks. *OR Spectrum*, 1989.

[6] M. Dorigo and T. Stützle. *Ant Colony Optimization.* MIT Press, 2004.

[7] H. Greenberg and F. Glover. New Approaches for Heuristic Search: A Bilateral Linkage With Artificial Intelligence. *European Journal of Operational Research*, 1989.

[8] F. Glover. Heuristics for Integer Programming Using Surrogate Constraints. *Decision Sciences*, 1977.

[9] S. Gupta and H. Luss. Allocation of Effort Resource Among Competing Activities. *Operations Research*, 1983.

[10] J. Holland. *Adaptation in Natural and Artifical Systems.* MIT Press, 1975.

[11] A. Mercier, J.-F. Cordeau and G. Laporte. A Unified Tabu Search Heuristic for Vehicle Routing Problems with Time Windows. *Journal of the Operational Research Society*, 2001.

[12] M. Johannesson and M. C. Weinstein. On the Decision Rules of Cost-Effectiveness Analysis. *Journal of Health Economics*, 12(4):459 – 467, 1993.

[13] M. Please, L. Lamport and R. Shostak. The Byzantine Generals Problem. *ACM Transactions on Programming Languages and Systems*, 1982.

[14] V. Maniezzo, M. Dorigo and A. Colorni. The Ant System: Ant Autocatalytic Pptimizing Process. *Technical Report Politenico di Milano*, 1991.

[15] G. Tecchiolli and R. Battiti. The Reactive Tabu Search. *ORSA Journal*, 1994.

[16] C. Gelatt, S. Kirkpatrick and M. Vecchi. Optimization by Simulated Annealing. *Science*, 1983.

[17] C. Scheideler. Network Flows III - Multicommodity Flows. University Lecture, 2003.

[18] N. Katoh, T. Ibarraki. *Resource Allocation Problems.* MIT Press, 1988.

[19] Google and Facebook Blocked by the Danish Child Pornography Filter. http://www.edri.org/edrigram/number10.5/danish-filter-blocks-google-facebook. last tested on 03.07.2013.

[20] Twitter is Blocked in Egypt Amidst Rising Protests. http://techcrunch.com/2011/01/25/twitter-blocked-egypt/. last tested on 03.07.2013.

[21] Y. Chang and Y.C. Hou. A New Efficient Encoding Mode of Genetic Algorithms for the Generalized Plant Allocation Problem. *Journal of Information Science and Engineering*, 2004.

[22] K. Poh, B. Yang, Y.S. Dai and M. Xie. Optimal Testing-Resource Allocation with Genetic Algorithm for Modular Software Systems. *The Journal of Systems and Software*, 2003.