Evolutionary Local Search for the Super-Peer Selection Problem and the p-Hub Median Problem

Steffen Wolf and Peter Merz

Distributed Algorithms Group
University of Kaiserslautern, Germany
{pmerz,wolf}@informatik.uni-kl.de

Abstract. Scalability constitutes a key property in Peer-to-Peer environments. One way to foster this property is the introduction of super-peers, a concept which has gained widespread acceptance in recent years. However, the problem of finding the set of super-peers that minimizes the total communication cost is NP-hard. We present a new heuristic based on Evolutionary Techniques and Local Search to solve this problem. Using actual Internet distance measurements, we demonstrate the savings in total communication cost attainable by such a super-peer topology. Our heuristic can also be applied to the more general Uncapacitated Single Assignment p-Hub Median Problem. The Local Search is then further enhanced by generalized *don't look bits*. We show that our heuristic is competitive with other heuristics even in this general problem, and present new best solutions for the largest instances in the well known Australia Post data set.

1 Introduction

During recent years *evolutionary algorithms* enhanced with *local search* have been used to solve many NP-hard optimization problems [1–4]. These heuristics take their power from the problem specific local search, while keeping all favorable features of the evolutionary approach.

We are especially interested in optimization problems connected with topology construction in Peer-to-Peer (P2P) systems. Well-known properties of these fully decentralized P2P systems include self-organizing and fault-tolerant behavior. In contrast to centralized systems, they usually possess neither a single point of failure nor other bottlenecks that affect the entire network at once. However, the scalability of such networks becomes an issue in the case of excessive growth: Communication times tend to increase and the load put on every node grows heavily when the networks get larger. A possible solution to this issue is the introduction of *super-peers*. Super-peers are peers that act as servers for a number of attached common peers, while at the same time, they form a network of equals among themselves. In a super-peer enhanced P2P network, each common peer is attached to exactly one super-peer, which constitutes its link to the remainder of the network. All traffic will be routed via the super-peers [5, 6].

To ensure smooth operation, the peers generally wish to maintain low-delay connections to the other peers. Hence, minimum communication cost is the aim when designing super-peer P2P networks. In this paper, we present a heuristic combining local search with evolutionary techniques for the Super-Peer Selection Problem (SPSP), i. e. the problem of finding the set of super-peers and the assignment of all other peers that minimizes the total communication cost.

Our special interest lies in the construction of these P2P overlay topologies. However, the problem of selecting the super-peers is strongly related to a hub location problem: the *Uncapacitated Single Assignment p-Hub Median Problem* (USApHMP) [7]. The USApHMP is a well known optimization problem and has received much attention in the last two decades. With minor adjustments, our heuristic can also be used for the USApHMP, which allows the comparison with other algorithms on established standard test cases.

This paper is organized as follows. In Section 2, we provide an overview of related work. In Section 3, we propose our Super-Peer Selection Heuristic. In Section 4, we present results from experiments on real world Internet data for the Super-Peer Selection Problem, as well as on standard test cases for the USApHMP, and compare the results with those of other recently published algorithms. The paper concludes with an outline for future research in Section 5.

2 Related Work

The Super-Peer Selection Problem, as proposed here, has not yet been studied in the literature. However, algorithms designed for the USApHMP can also be used for SPSP. The USApHMP has achieved much attention since it was presented by O'Kelly in [7], along with a set of test cases called CAB. Later, O'Kelly *et al.* also presented means of computing lower bounds for these problems [8]. Exact solutions have been computed by Ernst and Krishnamoorthy for problems with up to 50 nodes in [9]. In this paper, they also introduced a new test set called AP. Ebery presented two more efficient mixed integer linear programs (MILP) for the special case of only 2 or 3 hubs [10], and thus solved a problem with 200 nodes (2 hubs), and a problem with 140 nodes (3 hubs). Also, the authors of [9] presented a Simulated Annealing heuristic that found good solutions for problems with up to 200 nodes.

Skorin-Kapov *et al.* presented TabuHub [11], a heuristic method based on tabu search. Results were presented only for the smallest problems of the CAB set ($n \leq 25$). Also, neural network approaches have been proposed for the USApHMP. In [12], the memory consumption and the CPU time for these approaches was reduced. However, the neural network approach was again only applied to the smallest problems in the CAB set ($n \leq 15$). Unfortunately, no computation times are given, making comparisons with other heuristics difficult.

The most promising heuristic for the USApHMP so far has been presented by Pérez *et al.* in [13]. It is a hybrid heuristic combining Path Relinking [14] and Variable Neighborhood Search. The heuristic has proven to be very fast with both the CAB and AP sets, faster than any other heuristic. However, it

failed to find the optimum in some of the smaller CAB instances and still left room for improvements in the larger instances of the AP set. The local search neighborhoods used in this heuristic differ from the ones used here. Especially, the most expensive neighborhood is missing in [13]. This explains the speed as well as the loss of quality.

Two Genetic Algorithms have been presented by Kratica *et al.* [15]. These GAs are based on different representations and genetic operations. Both feature a mutation operator that favors the assignment of peers to closer super-peers, as well as a sophisticated recombination. The results of the second GA are the best results so far, as they improved the solutions for the larger AP instances found in [13]. However, the approach does not include a local search, and can still be improved. As far as we know, the heuristic we present in this paper is the first heuristic combining evolutionary techniques with local search.

3 Super-Peer Selection

When constructing a communication cost efficient and load balanced P2P topology we strive for a topology in which a subset of the nodes will function as super-peers while the rest of the nodes, henceforth called edge peers, is each assigned to one of the super-peers. Adhering to the established properties of super-peer overlay structures, the super-peers are fully connected among themselves and are able to communicate directly with the edge peers assigned to them and with their fellow super-peers. Essentially, the super-peers are forming the core of the network. The edge peers, however, will need to route any communication via their assigned super-peer. An example of such a super-peer topology is shown in Fig. 1. Using a topology of this kind, the communication between edge peers p_1 and p_{11} is routed via the super-peers c_1 and c_4. A broadcast in such a topology can be efficiently performed by having one super-peer send the broadcast to all other super-peers, which then forward the message to their respective edge peers. To ensure smooth operation and to ease the load on each peer, the number of super-peers should be limited as well as the number of peers connected to a super-peer.

The Super-Peer Selection Problem can be defined as finding the super-peer topology, i. e. the set of super-peers and the assignment of the edge peers to the super-peers, with minimal total communication cost for a given network. In a P2P setting, this cost can be thought of as the total all-pairs end-to-end communication delay.

3.1 Background

The SPSP is NP-hard [16]. It may be cast as a special case of the Hub Location Problem, first formulated by O'Kelly [7] as a Quadratic Integer Program. In the Hub Location Problem, a number of nodes, the so-called hubs, assume hierarchical superiority over common nodes, a property equivalent to the super-peer concept. Basically, given a network $G = (V, E)$ with $n = |V|$ nodes, p nodes

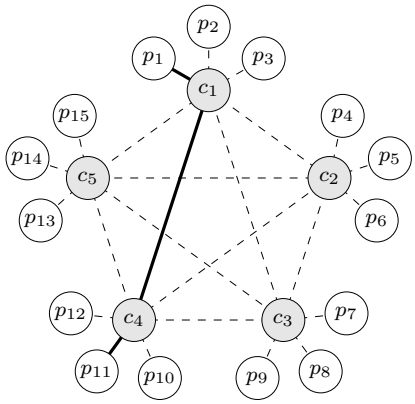


Fig. 1. Example of a P2P network with selected Super-Peers

are to be selected as hubs. Let x_{ik} be a binary variable denoting that node i is assigned to node k if and only if $x_{ik} = 1$. If $x_{kk} = 1$, node k is chosen as a hub. The flow volume between any two nodes $i \neq j$ is equal to one unit of flow. Since all flow is routed over the hubs, the actual weight on the inter-hub links is usually larger than one. The transportation cost of one unit of flow on the direct link between nodes i and j amounts to d_{ij}. Now, the SPSP formulated as a Hub Location Problem is

$$\min Z = \sum_{i=1}^{n} \sum_{j=1, j\neq i}^{n} \sum_{k=1}^{n} \sum_{m=1}^{n} (d_{ik} + d_{km} + d_{mj}) \cdot x_{ik} \cdot x_{jm} \tag{1}$$

s. t.

$$x_{ij} \leq x_{jj} \qquad i, j = 1, \ldots, n \tag{2}$$

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad i = 1, \ldots, n \tag{3}$$

$$\sum_{j=1}^{n} x_{jj} = p \tag{4}$$

$$x_{ij} \in \{0, 1\} \qquad i, j = 1, \ldots, n \tag{5}$$

Equation (1) yields the total communication cost Z. The set of constraints (2) ensures that nodes are assigned only to hubs, while (3) enforces the allocation of a node to exactly one hub. Due to constraint (4), there will be exactly p hubs.

A more general formulation uses a demand matrix $W = (w_{ij})$. Here, w_{ij} denotes the flow from node i to j in flow units. Also, special discount factors can be applied for the different edge types. Flow between hubs is subject to a discount factor $0 \leq \alpha \leq 1$, flow from a node to its hub is multiplied by a factor δ, and flow from a hub to a common node is multiplied by a factor χ. The total

communication cost Z is then:

$$Z = \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{m=1}^{n} (\chi \cdot d_{ik} + \alpha \cdot d_{km} + \delta \cdot d_{mj}) \cdot w_{ij} \cdot x_{ik} \cdot x_{jm} \tag{6}$$

The distinction between the different types of edges is motivated by an application in the area of mail transport. Here, the distribution cost differs from the collection cost. Also, the transportation cost between the hubs is assumed to be lower since more efficient means of transport can be used for the accumulated amount of flow. This extension might also be applied in the case of communication networks, especially when asymmetric links are considered. However, the most important difference from the SPSP is the introduction of demand factors w_{ij}, as will be shown in Section 3.3.

Since the objective function in both programs is quadratic and nonconvex, no efficient way to compute the minimum is known. The usual approach is to transform the problem into a Mixed Integer Linear Program (MILP). A straightforward linearization uses $\mathcal{O}(n^4)$ variables. We resort to an MILP formulation using as few as $\mathcal{O}(n^3)$ variables [17]:

$$\min Z = \sum_{i=1}^{n}\sum_{k=1}^{n} (\chi \cdot O_i + \delta \cdot D_i) \cdot d_{ik} \cdot x_{ik} + \sum_{i=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n} \alpha \cdot d_{kl} \cdot y_{ikl} \tag{7}$$

s. t. (2), (3), (4), (5),

$$\sum_{l=1}^{n} (y_{ikl} - y_{ilk}) = O_i \cdot x_{ik} - \sum_{j=1}^{n} w_{ij} \cdot x_{jk} \qquad i,k = 1,\ldots,n \tag{8}$$

$$y_{ikl} \geq 0 \qquad i,k,l = 1,\ldots,n \tag{9}$$

Here, $O_i = \sum_{j=1}^{n} w_{ij}$ is the outgoing flow for node i and $D_j = \sum_{i=1}^{n} w_{ij}$ is the demand of node j. Both values can be calculated directly from the problem instance. The variables y_{ikl} denote the flow volume from hub k to hub l which has originated at peer i. Constraints (8) and (9) ensure flow conservation at each node.

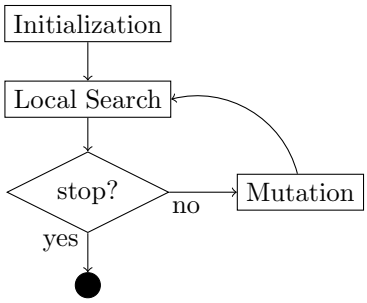
An MILP formulation for the SPSP can be derived by fixing $\chi = \delta = \alpha = 1$, $w_{ij} = 1$ for $i \neq j$, $w_{ii} = 0$, and thus $O_i = D_i = n - 1$:

$$\min Z = \sum_{i=1}^{n}\sum_{k=1}^{n} 2 \cdot (n-1) \cdot d_{ik} \cdot x_{ik} + \sum_{i=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n} d_{kl} \cdot y_{ikl} \tag{10}$$

s. t. (2), (3), (4), (5), (9),

$$\sum_{l=1}^{n} (y_{ikl} - y_{ilk}) = (n-1) \cdot x_{ik} - \sum_{j=1, j\neq i}^{n} x_{jk} \qquad i,k = 1,\ldots,n \tag{11}$$

The factor $2 \cdot (n-1)$ for the edge-peer to super-peer links in (10) is the number of connections using this link. It is based on the assumption that every edge peer



$s \leftarrow \textsc{Initialization}(p)$
$s \leftarrow \textsc{LocalSearch}(s)$
$\beta \leftarrow n$
while $\neg$ stopping criterion **do**
  for $i = 1 \dots m$ **do**
    $s_i \leftarrow \textsc{Mutation}(\beta, s)$
    $s_i \leftarrow \textsc{LocalSearch}(s_i)$
  end for
  $\min = \operatorname{argmin}_i\{Z(s_i)\}$
  if $Z(s_{\min}) < Z(s)$ **then**
    $s \leftarrow s_{\min}$
  else
    $\beta \leftarrow \max\{0.8 \cdot \beta, 2\}$
  end if
end while

Fig. 2. General overview of the Super-Peer Selection Heuristic

needs to communicate with all other $n - 1$ peers, and all other peers need to communicate with this edge peer.

These formulations are equivalent to the quadratic formulation only if the distances d_{ij} observe the triangle inequality. Otherwise, the model will generate solutions featuring the property that messages are sent along shortest paths between two hub instead of the intended direct link. The model can still be used for such networks. However, the resulting value can only serve as a lower bound.

The formulation above enables the exact solution of moderately-sized problems (up to 50 peers) in reasonable time, and additionally, the computation of lower bounds for larger networks (up to 150 peers) using its LP relaxation. For networks larger than the given threshold, we use the lower bounds described in [8]. Finally, the sum of all shortest paths' weights yields another lower bound.

3.2 Super-Peer Selection Heuristic

The Super-Peer Selection Heuristic presented here is based on *evolutionary algorithms* and *local search*. It operates on a global view of the network. The general work flow is shown in Fig. 2. The heuristic is quite similar to *iterated local search* [18], but uses more than one offspring solution in each generation.

Representation A solution is represented by the assignment vector s. For each peer i the value $s(i)$ represents the super-peer of i: $x_{i,s(i)} = 1$. All super-peers, the set of which will be denoted by C, are assumed to be assigned to themselves, i. e. $\forall i \in C : s(i) = i$. For the sake of swift computation, we also store the current capacities of the super-peers, i. e. the number of peers connected to the super-peer: $|V_k| = |\{i \in V \mid s(i) = k\}|$. This set also includes the super-peer itself: $k \in V_k$. The sets V_k are not stored explicitly, but are defined by the assignment vector s.

Initialization The initial solution is created by randomly selecting p peers as super-peers, and assigning all remaining peers to the nearest super-peer. When handling problems with missing links, this procedure is repeated if the initial set of super-peers is not fully connected.

Local Search After each step of the Evolutionary Algorithm a local search is applied to further improve the current solution. We use three different neighborhoods: *replacing the super-peer*, *swapping two peers* and *reassigning a peer to another super-peer*. If a neighborhood does not yield an improvement, the next neighborhood is used.

In the first neighborhood the local search tries to replace a super-peer by one of its children. The former child becomes the new super-peer and every other peer that was connected to the old super-peer is reconnected to the new super-peer. The gain of such a move can be computed in $\mathcal{O}(n)$ time. The following formula gives the gain for replacing super-peer k with i:

$$G_{\text{replace}}(i,k) = \sum_{j \in C} 2 \cdot |V_k| \cdot |V_j| \cdot (d_{kj} - d_{ij}) + \sum_{j \in V_k} 2 \cdot (n-1) \cdot (d_{kj} - d_{ij}) \quad (12)$$

If the gain of this move is $G_{\text{replace}}(i,k) > 0$, the move is applied.

The second neighborhood tries to exchange the assignment of two peers. The gain of such a move can be computed in $\mathcal{O}(1)$ time. Since all peers connected to other super-peers are considered as the exchange partner, the total time complexity for searching this neighborhood is $\mathcal{O}(n)$. Using the same notation as before, the following formula gives the gain for swapping the assignments of peers i and j:

$$G_{\text{swap}}(i,j) = 2 \cdot (n-1) \cdot (d_{i,s(i)} + d_{j,s(j)} - d_{i,s(j)} - d_{j,s(i)}) \quad (13)$$

The move with the highest gain is applied if its gain is $G_{\text{swap}}(i,j) > 0$.

The third neighborhood covers the reassignment of a peer to another super-peer. Here, it is important that the capacity limits of the involved super-peers are observed. The gain of reassigning peer i to super-peer k can be calculated in $\mathcal{O}(p)$ time:

$$\begin{aligned} G_{\text{reassign}}(i,k) = \quad & 2 \cdot (n-1) \cdot (d_{i,s(i)} - d_{i,k}) \\ & + 2 \cdot \big(|V_{s(i)}| \cdot |V_k| - (|V_{s(i)}| - 1) \cdot (|V_k| + 1) \big) \cdot d_{s(i),k} \\ & + 2 \cdot \sum_{j \in C \setminus \{k, s(i)\}} |V_j| \cdot (d_{s(i),j} - d_{k,j}) \end{aligned} \quad (14)$$

The first part of this equation gives the gain on the link between peer i and its super-peer. The second part gives the gain on the link between the old and the new super-peer. The third part gives the gain on all remaining intra-core links. Out of all super-peers only the one with the highest gain is chosen, thus yielding a total time complexity of $\mathcal{O}(p^2)$. The move is applied only if the total gain is $G_{\text{reassign}}(i,k) > 0$.

These local search steps are performed for each peer i. The local search is restarted whenever an improving step was found and applied. The local search is thus repeated until no improvement for any peer i can be found, i. e. a local optimum has been reached. Since all peers $i \in V$ are considered in these moves, the time complexity for searching the whole neighborhood is $\mathcal{O}(n^2)$.

Mutation Since local search alone will get stuck in local optima, we use mutation to continue the search. Mutation is done by swapping two random peers. Again, the "gain" of such a swap can be computed by (13). Several mutation steps are applied in each round. The number of mutations is adapted to the success rate. The algorithm starts with $\beta = n$ mutations. If no better solution is found in one generation, the mutation rate β is reduced by 20 %. In each round at least two mutations are applied. This way, the algorithm can adapt to the best mutation rate for the individual problem and for the phase of the search. It is our experience that it is favorable to search the whole search space in the beginning, but narrow the search over time, thus gradually shifting exploration to exploitation.

Population Our heuristic uses a population of only one individual. There is no need for recombination. This is mainly motivated by the high computation cost and solution quality of the local search. Using mutation and local search, m offspring solutions are created. The best solution is used as the next generation only if it yielded an improvement. This follows a $(1 + m)$ selection paradigm. If there was no improvement in the m children, the mutation rate β is reduced as described before.

Stopping criterion The heuristic is stopped after five consecutive generations without an improvement. This value is a compromise between solution quality and computation time. In the smaller instances the heuristic often finds the optimum in the first or second generation. Continuing the search would mean to waste time. We also stopped the heuristic after 40 generations regardless of recent improvements. Both values were chosen based on preliminary experiments.

3.3 Adaptation for the USApHMP

The USApHMP introduces weights w_{ij} on the connections between the nodes. While these weights have been equal for all node pairs in the Super-Peer Selection Problem, this is no longer the case in the full USApHMP. The main effect on the heuristic is that we can no longer summarize the flow on the inter-hub edges as $2 \cdot |V_a| \cdot |V_b|$. The following sum has to be used, instead: $\sum_{i \in V_a} \sum_{j \in V_b} w_{ij} + w_{ji}$. This would change the time complexity for calculating the cost of an inter-hub edge from $\mathcal{O}(1)$ to $\mathcal{O}(n^2)$. With the use of efficient data structures, however, the calculation for the cost of a move can be achieved in $\mathcal{O}(n)$ time.

Data structures In addition to the super-peers' capacities we also store the weights on the p^2 inter-hub links. $WC(a,b) = \sum_{i \in V_a} \sum_{j \in V_b} w_{ij}$ denotes the weight on the link from super-peer a to super-peer b. In each move made by the local search or the mutation these weights are changed accordingly. Only the selection of a new super-peer does not change these weights. Also, the gain calculations have to be adapted:

$$\begin{aligned} G_{\text{replace}}(i,k) = \alpha \cdot & \sum_{j \in C} (WC(j,k) + WC(k,j)) \cdot (d_{kj} - d_{ij}) \\ & + \sum_{j \in V_k} (\chi \cdot O_j + \delta \cdot D_j) \cdot (d_{kj} - d_{ij}) \end{aligned} \tag{15}$$

$$\begin{aligned} G_{\text{swap}}(i,j) = \alpha \cdot & \sum_{x \in V} (d_{s(j),s(x)} - d_{s(i),s(x)}) \cdot (w_{jx} - w_{ix} + w_{xj} - w_{xi}) \\ & + (\chi \cdot O_j + \delta \cdot D_j) \cdot (d_{j,s(j)} - d_{j,s(i)}) \\ & + (\chi \cdot O_i + \delta \cdot D_i) \cdot (d_{i,s(i)} - d_{i,s(j)}) \\ & - 2 \cdot \alpha \cdot d_{s(i),s(j)} \cdot (w_{ij} - w_{ii} + w_{ji} - w_{jj}) \end{aligned} \tag{16}$$

$$\begin{aligned} G_{\text{reassign}}(i,k) = \alpha \cdot & \sum_{x \in V} (d_{s(i),s(x)} - d_{k,s(x)}) \cdot (w_{ix} + w_{xi}) \\ & + (\chi \cdot O_i + \delta \cdot D_i) \cdot (d_{i,s(i)} - d_{ik}) + 2 \cdot \alpha \cdot d_{k,s(i)} \cdot w_{ii} \end{aligned} \tag{17}$$

The time complexity of calculating the gains is still $\mathcal{O}(n)$ for replacing the super-peer, but has increased to $\mathcal{O}(n)$ for swapping the assignments of two peers and to $\mathcal{O}(n)$ for reassigning a peer to another super-peer. Using the same local search as presented for the Super-Peer Selection Problem would mean to increase the total time complexity. We therefore implemented a reduced version of the most expensive local search: the swapping of two peers. Instead of calculating the gain for swapping one peer with all other $n - p - 1$ peers, we only calculate this gain for a random sample of p peers, which proved to be a good compromise between computation time and solution quality.

Don't look markers To further speed up the computation we use *don't look markers* to guide the local search. These markers are a generalization of *don't look bits*, that have been applied successfully for example to the *Traveling Salesman Problem* (TSP) [19]. In our algorithm, nodes that do not yield an improvement during one step of the local search will be marked. Nodes that are marked twice or more will not be checked in the following local search steps. However, if a node is part of a successful move all marks are removed again. This can happen if the node is the exchange partner of another node.

Using simple *don't look bits*, i. e. disregarding all nodes with one or more marks, leads to poor results, indeed. Here, too large parts of the considered neighborhoods will be hidden from the local search. Trying to reduce the negative impact of the *don't look bits* immediately lead to the more general *don't look markers*.

4 Evaluation

We have performed several experiments to study the effectiveness of our heuristic. For these experiments, we used real world node-to-node delay information from PlanetLab [20], a world-wide platform for performing Internet measurements. We used the round-trip times (RTT) from any host to any other host as the communication cost for the edges in the overlay network. From the measurements reported in [20] we used the first measurement for each month in 2005 (denoted by mm-2005). Those networks consist of $n = 70$ to $n = 419$ nodes. Common properties of all those networks are the frequent triangle inequality violations due to different routing policies and missing links most likely due to firewalls. We strive for $p \approx \sqrt{n}$ super-peers as a viable balance between low load and administrative efficiency. Also, the number x of common peers assigned to a super-peer is limited by $\frac{1}{2}\,p \leq x \leq 2\,p$. This ensures that no super-peer suffers from a high load, and that no peer is selected as super-peer without need.

For the USApHMP we used the CAB data set (Civil Aeronautics Board, up to 25 nodes) from [7, 21] and the AP data set (Australia Post, up to 200 nodes) from [9]. Both data sets have been widely accepted as standard test cases, and optima for all smaller instances are known. In the AP set the discount factors are fixed to $\alpha = 0.75$, $\chi = 3$ and $\delta = 2$. The CAB set fixes $\chi = \delta = 1$, but provides instances for different α-values: 0.2, 0.4, 0.6, 0.8 and 1.0. When refering to the individual instances in these sets, we will use the following notations: AP.n.p for the AP instance with n nodes and p hubs, CAB.n.p.α for the CAB instance with the corresponding configuration, and shorter CAB.n.p when $\alpha = 1.0$.

For each problem instance the heuristic was started 30 times and average values are shown. Computation times refer to a 2.8 GHz Pentium 4 with 512 MB RAM running Linux.

4.1 Lower bounds

For networks too large to be handled with the models described in Section 3.1, we are interested in computing lower bounds for SPSP. Table 1 contains the lower bounds for the networks considered here. The *all pairs shortest path* lower bound (APSP) yields the total communication cost (i. e. the sum of the distances of all node pairs) when communication is routed over shortest paths only. Column LB1 holds the lower bound defined in [8].

The column CPlex-LB holds an improved lower bound. We let CPlex 10.1 [22] solve the SPSP and find lower bounds. For the smallest network 04-2005, CPlex required eight days on a 3 GHz Pentium D with 4 GB RAM to arrive at a gap of 1.55 %. For network 01-2005, we stopped CPlex after it has consumed three weeks of CPU time on the same machine. For all other networks, CPlex required an exceedingly long period of time, hence was unable to provide viable results.

4.2 Results for the Super-Peer Selection Problem

In Table 2, the best solutions ever found by our heuristic are compared with the lower bounds from the previous table. This also includes runs of the heuristic

Network	Size	p	APSP	LB1	CPlex-LB
01-2005	127	12	2 447 260	2 501 413	2 632 988
02-2005	321	19	15 868 464	16 200 776	
03-2005	324	18	17 164 734	17 580 646	
04-2005	70	9	663 016	690 888	728 576
05-2005	374	20	17 324 082	17 794 876	
06-2005	365	20	18 262 984	18 735 944	
07-2005	380	20	24 867 734	25 337 991	
08-2005	402	21	27 640 136	28 151 142	
09-2005	419	21	23 195 568	23 646 759	
10-2005	414	21	28 348 840	28 905 539	
11-2005	407	21	23 694 130	24 196 510	
12-2005	414	21	20 349 436	20 885 442	

Table 1. Different lower bounds for the considered networks

with higher numbers of offspring and relaxed stopping criterion (500 offsprings, 1000 generations, different β adaptation, no stopping after five consecutive generations without improvement). There is still a considerable gap between the LB1 lower bounds and these best known solutions, ranging from 7.1 % to 44.5 %. We believe these best known solutions to be close to the optimum, though, since the LB1 lower bound is too low in instances with many triangle inequality violations. In fact, the best known solutions for both 04-2005 and 01-2005 are still within the lower and upper bounds found by CPlex.

The comparison of these best known solutions with unoptimized super-peer topologies quantifies the benefit of optimization. The column Gain in Table 2 gives the quotient of an average random configuration's cost to the best known solution's cost. Overlay topologies when constructed without locality awareness can be assumed to be random. Accordingly, the communication cost in real world networks becomes subject to reduction by a factor of 2.5 to 3.8 compared to the unoptimized solution, and even the smallest network's total communication cost could still be successfully optimized by a factor of 2.5 compared to its unoptimized counterpart. The high gain of 7.3 in network 12-2005 yields from the large extent of triangle inequality violations in this network.

For the actual parameter settings as described in Section 3.2, Table 3 shows the average excess over the best known solutions, the CPU times per run and the success rate as the number of runs that found the best known solutions. These results show that the heuristic is able to find solutions close to the best known solutions in all runs. The average excess is never higher than 1.1 %. Even with the tight stopping criterion and the reduced number of offspring the heuristic finds the best known solution in some cases. The CPU times depend on the size of the network. The heuristic could be stopped earlier, but this would result in worse solution quality.

Unfortunately, the results for the SPSP can not be directly compared to other heuristics. However, with the more general USApHMP and established standard

Network	Best known	Excess over LB1	Gain
01-2005	2 929 830	17.1 %	3.8
02-2005	18 620 614	14.9 %	2.7
03-2005	20 715 716	17.8 %	2.7
04-2005	739 954	7.1 %	2.5
05-2005	25 717 036	44.5 %	2.8
06-2005	22 319 228	19.1 %	3.1
07-2005	31 049 398	22.5 %	3.2
08-2005	30 965 218	10.0 %	3.1
09-2005	33 039 868	39.7 %	3.2
10-2005	32 922 594	13.9 %	3.4
11-2005	27 902 552	15.3 %	3.3
12-2005	28 516 682	36.5 %	7.3

Table 2. Best known solutions, their excess over the LB1 lower bound and the gain compared to random configurations for the considered networks

test sets we can show that our heuristic is competitive with the algorithms proposed in the literature.

4.3 Results for the USApHMP

Since the full USApHMP is more complex than the SPSP, we use the adapted heuristic as described in Section 3.3 for these experiments. The AP set consists of 20 smaller ($n \leq 50$) and 8 larger instances ($n \geq 100$). Optimal solutions are known only for the smaller instances. The CAB set consists of 60 instances (four different sizes up to 25 nodes, three different numbers of hubs, five different discount factors α). For all these instances the optimum is known. Again, each experiment was repeated 30 times.

In all 600 runs for the smaller instances of the AP set our heuristic reached the known optimum. Only in six out of all 1800 runs with the CAB set the optimum was not reached. Table 4 gives details on these runs. In all these cases a less strict stopping criterion helps to reach the optimum again.

This very good solution quality can not be kept up on the larger instances of the AP set. Only in 51 out of the 240 runs on these instances the best known solution was reached. However, the average excess above those best known solutions is still considerably good, as Table 5 shows. The best solutions known so far for the larger instances of the AP set have been listed in [15]. Our heuristic is able to reach these solutions in all but one problem (AP.200.20), while taking roughly the same CPU time as [15]. We have also found new best solutions for two cases (AP.200.5 and AP.200.15). These new best solutions are marked in Table 5. Our heuristic seems to be more effective in finding the best solutions for problems with less hubs. For example the heuristic never failed to find the best solution in AP.100.5 and also often finds the new best solution for AP.200.5.

Network	Excess over best known	CPU time per run	# best found
01-2005	0.49 %	13.6 s	1/30
02-2005	0.72 %	241.8 s	1/30
03-2005	0.69 %	176.9 s	1/30
04-2005	0.38 %	1.5 s	2/30
05-2005	1.06 %	302.0 s	0/30
06-2005	0.55 %	275.7 s	1/30
07-2005	0.66 %	299.2 s	1/30
08-2005	1.00 %	286.6 s	0/30
09-2005	1.05 %	370.8 s	0/30
10-2005	1.06 %	384.4 s	0/30
11-2005	0.83 %	368.9 s	0/30
12-2005	0.78 %	353.8 s	1/30

Table 3. Average excess over best known solution and CPU times

Instance			Excess over	# optimum
n	p	α	optimum	found
15	2	0.6	0.038 %	29/30
15	3	1.0	0.044 %	28/30
20	4	1.0	0.004 %	28/30
25	4	1.0	0.040 %	29/30
all other			—	30/30

Table 4. Average excess above the optimum for the CAB set

For problems with more hubs the success rate decreases and the average excess over the best known solution increases.

Also, since the search is more complex for problems with more hubs, the average CPU time increases, as Table 6 shows. The computation times range from less than a second for all smaller instances to about seven minutes for the largest one in the AP set. All computation times for the CAB set are well below one second and seem independent on the intra-core discount factor α. The larger the problem instance and the more hubs are to be located the more time the heuristic uses. This behavior can also be observed for other heuristics and is therefore not surprising.

5 Conclusion

We have presented a hybrid method combining evolutionary algorithms and local search for the Super-Peer Selection Problem and the USApHMP. This heuristic has proven to find optima in all smaller USApHMP instances. The heuristic uses

Instance n	p	Excess over best known solution	# best found	Best known cost	
100	5	0.00 %	30/30	136 929.444	
100	10	0.30 %	5/30	106 469.566	
100	15	0.75 %	1/30	90 533.523	
100	20	1.62 %	1/30	80 270.962	
200	5	0.16 %	11/30	140 062.647	improved
200	10	0.17 %	1/30	110 147.657	
200	15	0.72 %	2/30	94 459.201	improved
200	20	1.35 %	0/30	85 129.343	

Table 5. Average excess above the best known solutions for the AP set for $n \geq 100$

n	$p = 2$	$p = 3$	$p = 4$	$p = 5$	$p = 10$	$p = 15$	$p = 20$
10	0.07	0.08	0.09	0.09			
20	0.14	0.20	0.24	0.28			
25	0.19	0.29	0.37	0.45			
40	0.37	0.64	0.99	1.48			
50	0.58	0.94	1.42	2.14			
100				11.19	25.81	63.01	77.87
200				58.10	188.05	305.05	417.41

Table 6. Average CPU times per run in seconds for the AP set

a more thorough search than previous algorithms, and so has found new best solutions for two of the largest instances in the AP set ($n = 200$).

The time complexity improves when applying the heuristic to the Super-Peer Selection Problem. Here, we are able to tackle problems with $n = 400$ and more nodes within reasonable time. The results show that the heuristic is able to optimize the total communication costs in unoptimized real world super-peer topologies by a factor of about 3.

We are also working on a distributed algorithm to solve the SPSP using only local view of the involved peers. First results are already promising. This algorithm will be integrated into a middleware for Peer-to-Peer Desktop Computing.

References

1. Merz, P., Freisleben, B.: Fitness Landscapes and Memetic Algorithm Design. In Corne, D., Dorigo, M., Glover, F., eds.: New Ideas in Optimization. McGraw–Hill, London (1999) 245–260
2. Merz, P.: Memetic Algorithms for Combinatorial Optimization Problems: Fitness Landscapes and Effective Search Strategies. PhD thesis, Department of Electrical Engineering and Computer Science, University of Siegen, Germany (2000)
3. Hoos, H.H., Stützle, T.: Stochastic Local Search: Foundations and Applications. The Morgan Kaufmann Series in Artificial Intelligence. Morgan Kaufmann (2004)

4. Merz, P., Fischer, T.: A Memetic Algorithm for Large Traveling Salesman Problem Instances. In: MIC'2007 – 7th Metaheuristics International Conference. (2007)
5. Yang, B., Garcia-Molina, H.: Designing a super-peer network. In: Proceedings of the 19th International Conference on Data Engineering. (2003) 49–62
6. Li, D., Xiao, N., Lu, X.: Topology and resource discovery in Peer-to-Peer overlay networks. In: Grid and Cooperative Computing – GCC 2004 Workshops. Volume 3252 of Lecture Notes in Computer Science., Springer (2004) 221–228
7. O'Kelly, M.E.: A quadratic integer program for the location of interacting hub facilities. European Journal of Operational Research **32** (1987) 393–404
8. O'Kelly, M.E., Skorin-Kapov, D., Skorin-Kapov, J.: Lower bounds for the hub location problem. Management Science **41**(4) (1995) 713–721
9. Ernst, A.T., Krishnamoorthy, M.: Efficient algorithms for the uncapacitated single allocation p-hub median problem. Location Science **4**(3) (1996) 139–154
10. Ebery, J.: Solving large single allocation p-hub problems with two or three hubs. European Journal of Operational Research **128**(2) (2001) 447–458
11. Skorin-Kapov, D., Skorin-Kapov, J.: On tabu search for the location of interacting hub facilities. European Journal of Operational Research **73** (1994) 502–509
12. Domínguez, E., Muñoz, J., Mérida, E.: A recurrent neural network model for the p-hub problem. In Mira, J., Álvarez, J.R., eds.: IWANN 2003: International Work-Conference on Artificial and Natural Neural Networks. Volume 2687 of Lecture Notes in Computer Science., Springer (2003) 734–741
13. Pérez, M.P., Rodríguez, F.A., Moreno-Vega, J.M.: A hybrid VNS-path relinking for the p-hub median problem. In: IMA Journal of Management Mathematics, Oxford University Press (2007)
14. Pérez, M.P., Rodríguez, F.A., Moreno-Vega, J.M.: On the use of path relinking for the p-hub median problem. In Gottlieb, Raidl, eds.: Proceedings of the 7th European Conference on Evolutionary Computation in Combinatorial Optimization. Volume 3004 of Lecture Notes in Computer Science., Springer (2004) 155–164
15. Kratica, J., Stanimirović, Z., Tošić, D., Filipović, V.: Two genetic algorithms for solving the uncapacitated single allocation p-hub median problem. European Journal of Operational Research **182**(1) (2007) 15–28
16. Wolf, S.: On the complexity of the uncapacitated single allocation p-hub median problem with equal weights. Internal Report 363/07, University of Kaiserslautern, Kaiserslautern, Germany (2007) Available at `http://dag.informatik.uni-kl.de/papers/Wolf2007SPSP-NP.pdf`.
17. Ernst, A.T., Krishnamoorthy, M.: An exact solution approach based on shortest-paths for p-hub median problems. INFORMS Journal on Computing **10**(2) (1998) 149–162
18. Lourenço, H.R., Martin, O., Stützle, T.: Iterated Local Search. In Glover, F., Kochenberger, G., eds.: Handbook of Metaheuristics. (2002) 321–353
19. Bentley, J.L.: Experiments on traveling salesman heuristics. In: SODA'90: Proceedings of the first annual ACM-SIAM symposium on Discrete algorithms, Philadelphia, PA, USA, Society for Industrial and Applied Mathematics (1990) 91–99
20. Banerjee, S., Griffin, T.G., Pias, M.: The Interdomain Connectivity of PlanetLab Nodes. In Barakat, C., Pratt, I., eds.: Proc. of the 5th International Workshop on Passive and Active Network Measurement. (2004)
21. O'Kelly, M.E., Bryan, D.L., Skorin-Kapov, D., Skorin-Kapov, J.: Hub network design with single and multiple allocation: A computational study. Location Science **4**(3) (1996) 125–138
22. ILOG S.A.: ILOG CPLEX User's Manual (2006) Gentilly, France, and Mountain View, California. `http://www.cplex.com/`.