Evolutionary Local Search for Designing Peer-to-Peer Overlay Topologies based on Minimum Routing Cost Spanning Trees

Peter Merz and Steffen Wolf*

Department of Computer Science
University of Kaiserslautern, Germany
{pmerz,wolf}@informatik.uni-kl.de

Abstract. Finding overlay topologies for peer-to-peer networks on top of the Internet can be regarded as a network design problem, in which a graph with minimum communication costs is desired. An example of such a graph is a spanning tree connecting all nodes in the overlay. We present evolutionary algorithms incorporating local search for the minimum routing cost spanning tree problem in which the overall routing/communication cost is minimized. We present three types of local search for this problem as well as an evolutionary framework for finding (near)optimal solutions to the problem. Moreover, we present results from a fitness landscape analysis for the three types of local optima that reveal interesting properties of the problem data based on real measurements in the Internet. We demonstrate that our proposed algorithms find near optimum solutions reliably by comparing against a lower bound of the problem.

1 Introduction

Peer-to-peer (P2P) systems have become popular due to development of file-sharing applications like Gnutella [1, 2], Kazaa or Bit-Torrent [3]. However, the P2P paradigm is useful in many other application scenarios such as Distributed Computing [4, 5], Distributed Storage/Backup [6], or Distributed Database Systems [7]. An important aspect in the development of a P2P protocol is the design of the overlay topology. The topology defines which nodes are directly connected to each other. More formally, we can think of the topology as a graph, in which the nodes are networked computers and the edges are communication links between them. The goal of the design is to find a graph which is robust in respect to failures of nodes and links and/or which reduces the communication overhead and time within the network. In this paper, we concentrate on the latter by minimizing the routing and hence communication time between any pair of nodes in the network. Since the graph should be easy to maintain as well as to repair and routing algorithms should be simple, we focus on spanning trees.

* This work was partially supported by the Rhineland-Palatinate Cluster of Excellence 'Dependable Adaptive Systems and Mathematical Modelling'.

Since the forming of the topology is a self-organizing process, we are interested in distributed online algorithms for the network design problem. However, we concentrate in this paper on the offline case, where we search for the optimum topology given the global view on the network. The purpose of this approach is twofold. Firstly, the offline algorithms provide a basis for analyzing the performance of online algorithms. Secondly, by analyzing local search algorithms we gain valuable insight for the development of distributed algorithms for the online optimization problem.

The paper is organized as follows. In section 2, an overview is given for network optimization problems with focus on the minimum routing cost spanning tree (MRT) problem as well as local search algorithms. A new evolutionary heuristics for the MRT problem is presented in section 3. In section 4, results from various experiments with the new heuristics as well as a landscape analysis are presented based on measured Internet (PlanetLab) data. Conclusions and an outline for future research is provided in section 5.

2 Network Optimization and Local Search

A P2P overlay topology can be represented by a weighted graph $G = (V, E, d)$ where the nodes are the peers and the edges are communication links between the peers used for communicating in the P2P system. The weight $d(i, j)$ of an edge (i, j) denotes message delay on the communication link from peer i to peer j, or the average round-trip-time (RTT) between peer i to peer j. A spanning tree is a subgraph $T \subseteq G$ with $|V| - 1$ edges connecting all nodes in V. Given a spanning tree, multicast/broadcast can be implemented simply by flooding. In this preliminary work, we focus on spanning trees as topologies for P2P overlays since they require a minimum number of edges (connections) and are easily maintained. Since in a P2P system peers may join or leave at any time, the graph is changing over time. In order to study the effectiveness of the optimization methods, we concentrate on the case where the graph remains constant for the time of optimization.

2.1 Problem Definition

In order to minimize the time for multicast or unicast routing, a tree with optimum routing cost is sought. The problem of finding a spanning tree with minimum routing cost is defined as follows. Given a graph $G = (V, E, d)$, finding a spanning tree T of G such that

$$C(T) = \sum_{u,v \in V} d_T(u, v) \tag{1}$$

is minimal is called the *Minimum Routing Cost Spanning Tree (MRT) Problem*, where $d_T(u, v)$ denotes the distance/length of the path from u to v in T (message delay from u to v). The problem is a special case of the optimum communication

spanning tree problem, which is known to be NP-hard [8, 9], but known to admit a polynomial-time approximation scheme (PTAS) [9, 10].

The MRT problem is closely related to the Shortest-Path-Tree (SPT) Problem. In fact, there is a vertex such that any SPT rooted at the vertex is a 2-approximation of the MRT. The SPT problem can be solved in polynomial time with Dijkstra's algorithm [11] or the algorithm of Bellman and Ford [12, 13]. However, resulting trees may degenerate to stars if the triangle inequality is obeyed.

2.2 Related Work

Only few research has been done on the optimization of network topologies, in particular overlay topologies, focusing on combinatorial optimization techniques. In [14], two combinatorial problems are addressed, a minimum spanning tree problem, and a traveling salesman problem. In both cases, centralized heuristics/algorithms are proposed. In [15], evolutionary algorithms for self-organized networks are proposed. However, the authors do not consider the MRT objective and it is unclear how their algorithms can work in a fully decentralized distributed system. A centralized approximation algorithm for application-layer multicast trees is presented in [16]. The approach concentrates on an NP-hard optimization problem in which each node has a nonnegative processing delay the model differs also from our problem regarding the objective. The MeshTree approach [17] is a decentralized approach differentiating between backbone and delivery trees. Each node has a maximum neighbor degree. It is unclear how effective the optimization is in terms of approaching the optimum or a lower bound. Moreover, the benefits from using a minimum spanning tree (MST) as a backbone tree are not obvious. Since finding a degree-constrained MST is an NP-hard problem it can be expected to be harder for heuristic search than the degree-constrained SPT. Summarizing, to the best of our knowledge the MRT problem has not yet been addressed by (evolutionary) heuristics.

2.3 Local Search for the MRT Problem

Local search in the MRT Problem works by searching for a better solution (in terms of fitness/cost function) in the neighborhood of the current solution. The neighborhood itself consists of all trees reachable from a given tree by applying simple 'moves'. Such a move modifies the tree in a simple non-destructive way. Two types of moves considered in our algorithms are shown in Fig. 1. In the first move (a), a node (i) is connected to a new parent (q). We denote a neighborhood based on this move a 1-opt neighborhood. The second move (b) is actually a combination of two moves of type (a): two nodes, denoted i and j exchange their parents. We denote the neighborhood of this move a 2-opt neighborhood.

In order to compute the gain associated with a move, we make use of the following formula. The objective can be rewritten as

$$C(T) = \sum_{v \in V} C_v(T) \quad \text{with } C_v(T) = 2\, c(v)\, (N - c(v))\, d(v, p^v), \tag{2}$$

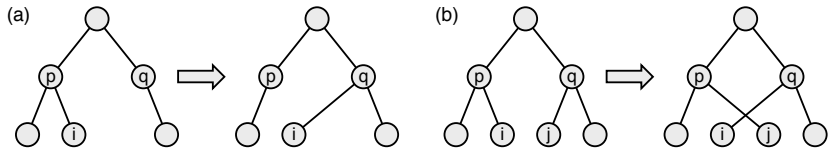

Fig. 1. Two types of tree moves

where $N = |V|$, p^v denotes the parent of node v and $c(v)$ denotes the size of the subtree rooted at v (number of children + 1). We assume here that the tree is rooted with an arbitrary root. With the formula, the cost of a tree can be computed in linear time. Also, this formula can be used to calculate the gain of a move efficiently. For both moves, only those C_v have to be recalculated that do change. These are those nodes for which the number of children or parent edges change and hence all nodes on the path P on the tree from i to parent q (or node j):

$$\Delta C(T \to T') = C(T) - C(T') = \sum_{v \in P} (C_v(T) - C_v(T')) \tag{3}$$

Hence, the number of updates is expected to be in $O(\log N)$ but in the worst case linear in N (if the tree degenerates to a single path). If we except an expected path length of $O(\log N)$, the time to search the complete neighborhood (denoted by $N_{2\text{-opt}}$) becomes $O(N^2 \log N)$. As a consequence, a local search based on these neighborhoods may not scale well with the problem size. Therefore, we propose two variants of the local search.

The first one considers only the nodes incident to the parent node p as candidates for the new parent q in move (a). Hence, the path from p to q consists of just p and q. A move of type (b) is only performed if $c(i) = c(j)$ and hence no update of C_v is needed along the path from p to q. Furthermore, not all pairs (i, j) are considered here. Instead, in each iteration of the local search, for each node i a node j is selected randomly. Therefore, the time complexity for searching the neighborhood reduces to $O(N)$ if we assume that the node degree in the tree is bounded by a constant. In the following we refer to this neighborhood as the constant update neighborhood N_{const}.

Since a local search based only on constant time/local changes may get stuck too early in local optima, we considered another neighborhood. In addition to N_{const} we also check for non-local moves of type (a) by randomly selecting j for each node i. This way, the time required for searching the neighborhood remains sub-quadratic but global moves are also considered to a certain degree. We call this neighborhood N_{fast}.

3 An Evolutionary Algorithm for the MRT Problem

Since the local search alone may not be sufficient to find optimum or near optimum solutions, we propose an evolutionary framework for improving local opti-

mum solutions. The framework is similar to *iterated local search* [18] or *memetic algorithms* [19, 20]. Unlike other memetic algorithms for combinatorial problems, it uses only mutation.

Since the general framework is independent of the considered problem, we provide a problem independent description in the following.

3.1 The General Evolutionary Framework

After creating initial solutions or an application of the mutation operator, a local search is applied to find a new local optimum. The general outline is shown in Fig. 2.

The framework is especially useful if not much is known about a problem. It accepts three parameters: the number of generations κ, the number of offspring per iteration λ, and the mutation adaptation rate α. The algorithm can be considered as a $(1+\lambda)$-EA incorporating local search. The product of generations and offspring defines the number of local searches to perform. The higher the number of offspring the higher the selection pressure. The adaptation rate defines by which factor the number of mutation steps is to be reduced if an iteration was not successful, i. e. there was no better solution. This approach is meaningful if the optimum mutation strength of problem is not known. Especially if an optimum mutation rate exists but is unknown, the adaptation allows to try different mutation rates or no mutation at all if multiple starts (init+local search) are sufficient. Since the number of mutation steps is adapted only if there was no improvement, the algorithm is capable of selecting an effective mutation rate depending on the state of convergence.

```
function EALS( κ: Integer; λ: Integer; α : Real) : Solution;
  begin
    s := Init ();
    s := LocalSearch(s);
    mutationSteps := ProblemSize(s);
    for iter := 0 to κ do begin
      sbest := s;
      for i := 0 to λ do begin
        if mutationSteps = ProblemSize(s) then stemp := Init()
          else stemp := Mutate(s, mutationSteps);
        stemp := LocalSearch(stemp);
        if stemp < sbest then sbest := stemp;
      end;
      if sbest < s then s := sbest
        else mutationSteps := round(mutationSteps ∗ α);
    end;
    return s;
  end;
```

Fig. 2. The Evolutionary Local Search Framework

3.2 The Local Search Operator for the MRT Problem

It is not obvious which local search is best suited in the above evolutionary framework. Therefore, we decided to realize all three local search procedures described above, namely N_{const} local search, N_{fast} local search and $N_{\text{2-opt}}$ local search.

3.3 The Mutation Operator for the MRT Problem

The mutation operator is parameterized by a the number of mutation steps to apply. A single step consists of a random move as used in the local search, more precisely, type (a) described above. Hence, within the mutation operator such a move with a randomly selected node and a randomly selected new parent is applied k times with k denoting the number of mutation steps (mutationSteps in the pseudo code). As a consequence of mutation with k steps will remove k edges from the tree and insert k random edges such that the resulting graph is again a tree.

4 Results

We performed several experiments to evaluate the effectiveness of the local search variants as well as the evolutionary framework. In the experiments, we used data collected in experiments performed on the PlanetLab [21], a world-wide platform for performing Internet measurements. The PlanetLab consist of more than 400 computers distributed over more than 200 sites around the world. The measurements are simple 'ping' measurements measuring the RTT of messages using the ICMP protocol from any host to any other host in the PlanetLab. These RTTs were used in our experiments as the communication cost for the edges in the overlay graph. The higher the RTT, the higher the cost for a link/edge. In the experiments, we used the daily collected data by Jeremy Stribling reported in [21]. For each month of the year, we selected the first measurement (denoted by 01-2005, ..., 12-2005). All CPU times reported in this section refer to a Pentium IV (3 GHz, running Java). All results are averaged over at least 30 runs.

4.1 EALS Performance Evaluation

In a first set of experiments, we ran our evolutionary algorithms with varying parameters on the set of 12 instances. We varied the number of generations κ, the number of offspring λ, and the local search variant used in the EA. In all experiments we fixed the adaptation rate to $\alpha = 0.8$. The results for some of the experiments are shown in Table 1. In the displayed experiments the number of offspring was set to 1, and the number of generations to 1000. The cost of the best known solution is shown in column Best. It can be seen that with increasing neighborhood size $|N_{\text{const}}| < |N_{\text{fast}}| < |N_{\text{2-opt}}|$ both runtime and solution quality (shown as the percentage excess over the best known solution) increase. The constant update neighborhood local search performs very poor on some of the instances with a high deviation of more than 27 % from the best solution.

		(1000,1,const)		(1000,1,fast)		(1000,1,2-opt)	
Instance	Best	Time	Excess	Time	Excess	Time	Excess
01-2005	2743556	0.9 s	3.86	1.4 s	1.33	37.2 s	0.26
02-2005	17567152	3.6 s	8.86	5.7 s	1.46	313.1 s	0.34
03-2005	19288320	3.5 s	10.02	5.6 s	1.72	286.2 s	0.52
04-2005	751404	0.5 s	1.03	0.7 s	0.50	8.5 s	0.09
05-2005	19175890	4.4 s	39.06	7.6 s	1.58	513.6 s	0.17
06-2005	20312884	4.1 s	16.27	7.1 s	1.61	409.1 s	0.29
08-2005	30540984	4.9 s	6.78	7.9 s	0.86	542.9 s	0.26
09-2005	25712960	5.3 s	30.76	8.7 s	1.58	535.4 s	0.26
11-2005	26797284	5.3 s	9.39	8.5 s	0.62	527.2 s	0.17

Table 1. Results for three Evolutionary Locals Search algorithm variants, with $(\kappa, \lambda, LS) = (1000, 1, \{\text{const}, \text{fast}, \text{2-opt}\})$, κ denoting the number of generations, λ the number of offspring per generation, and LS the local search neighborhood

In order to study the influence of the number of generated offspring per generation, we performed additional experiments. The most relevant results are summarized in Table 2. The time required (Time) as well as percentage excess over the best known solution in percent (Ex) are reported. Here, we can see that increasing the number of offspring significantly increases average solution quality. Moreover, the N_{fast} local search EA appears to be close to the $N_{\text{2-opt}}$ local search EA in terms of tree cost, given approximately the same CPU time. The results clearly indicate that the 2-opt local search EA performs the best but the fast local search EA can achieve also very good results. It may scale better with the problem size if the networks become larger than investigated here. The slight modification of the fast variant compared to the const variant has a tremendous effect on the solution quality for several instances considered. The 2-opt variant with 100 generations and 100 offspring per generation finds the best solutions very frequently as indicated in the last column of the table.

4.2 Problem Instance Analysis

In the next set of experiments, we looked at the deviation of our best solutions found from two bounds, the all-pairs-shortest-path (APSP) lower bound and the best-shortest-path-tree (SPT) upper bound. The former measures the overall communication cost if we consider for each node the shortest path tree instead of a single shared tree as in the MRT. The latter is the best shortest path tree among all possible shortest path trees in terms of our cost function. This tree is a 2-approximation to the optimum solution. The results are shown in Table 3. Interestingly, it turns out that there is a relatively small gap of 10 % to 13 % between the best solutions we found and the simple APSP lower bound for all instances independent of their size. The gap to the upper bound of 4 % up to 14 % indicates that it is worth using heuristics for the MRT problem instead of using the shortest path tree approximation.

In a final set of experiments, we were interested in the properties of the local optima using the three neighborhoods. We calculated the average percentage ex-

		(100,100,fast)		(10,10,2-opt)		(100,100,2-opt)		
Instance	Best	Time	Ex	Time	Ex	Time	Ex	Best
01-2005	2743556	18.2 s	0.29	9.9 s	0.13	157.6 s	0.00	52/53
02-2005	17567152	108.7 s	0.28	97.6 s	0.20	3260.0 s	0.00	34/50
03-2005	19288320	105.3 s	0.22	92.5 s	0.07	2633.5 s	0.00	42/48
04-2005	751404	5.4 s	0.05	2.0 s	0.01	26.6 s	0.00	48/48
05-2005	19175890	166.2 s	1.17	155.7 s	0.08	3244.8 s	0.00	43/48
06-2005	20312884	148.8 s	0.26	132.2 s	0.07	5558.5 s	0.00	34/48
08-2005	30540984	162.7 s	0.31	168.5 s	0.12	4067.4 s	0.00	39/47
09-2005	25712960	195.0 s	0.25	185.2 s	0.03	3840.2 s	0.00	35/48
11-2005	26797284	178.1 s	0.16	177.9 s	0.03	6407.2 s	0.00	40/48

Table 2. Results for three Evolutionary Locals Search algorithm variants, with $(\kappa, \lambda, LS) = \{(100, 100, \text{fast}), (10, 10, \text{2-opt}), (100, 100, \text{2-opt})\}$, κ denoting the number of generations, λ the number of offspring per generation, and LS the neighborhood used in the local search

Instance	Size	Best	Lower Bound (APSP)	Upper Bound (SPT)
01-2005	127	2743556	2447260 (12.1%)	3025120 (10.3%)
02-2005	321	17567152	15868464 (10.7%)	19607936 (11.6%)
03-2005	324	19288320	17164734 (12.4%)	20842606 (08.1%)
04-2005	70	751404	663016 (13.3%)	787856 (04.9%)
05-2005	374	19175890	17324082 (10.7%)	19949036 (04.0%)
06-2005	365	20312884	18262984 (11.2%)	22217312 (09.4%)
08-2005	402	30540984	27640136 (10.5%)	32209226 (05.5%)
09-2005	419	25712960	23195568 (10.9%)	27223336 (05.9%)
11-2005	407	26797284	23694130 (13.1%)	29322480 (09.4%)

Table 3. Overview of all considered instances. For each instance, the network size, the best solution found, the All-Pairs-Shortest-Path lower bound and an upper bound based on the Best Shortest Path Tree are shown.

	LS const			LS 2-opt			LS fast		
Instance	Ex(%)	FDC	Dist	Ex(%)	FDC	Dist	Ex(%)	FDC	Dist
01-2005	68.8	0.25	91.3	5.5	0.32	52.7	13.7	0.40	80.3
02-2005	51.5	0.24	237.0	4.1	0.43	134.7	12.0	0.46	207.6
03-2005	47.0	0.22	245.6	3.6	0.37	145.5	11.3	0.40	216.2
04-2005	21.0	0.33	41.2	2.9	0.28	27.2	7.2	0.30	35.6
05-2005	86.9	0.15	282.9	3.0	0.39	143.5	32.7	0.50	245.3
06-2005	51.2	0.23	274.0	3.8	0.49	144.8	14.8	0.49	235.9
08-2005	34.1	0.31	291.5	2.8	0.48	158.3	8.0	0.52	250.9
09-2005	73.4	0.27	311.0	3.4	0.36	164.9	29.6	0.45	270.9
11-2005	43.3	0.31	293.8	3.2	0.45	156.9	10.2	0.44	255.0

Table 4. Properties of the local optima. For each local search neighborhood, the deviation from the best solution, the fitness distance correlation coefficient, and the average distance to the best solution is provided.

cess (Ex) over the best solution found, the fitness distance correlation coefficient (FDC) of the local optima and the average distance (Dist) in the solution space to the best known solution. This distance is calculated by counting the edges of one tree not contained in the other. The results are displayed in Table 4. It can be observed that average tree quality and correlation are significantly lower for the simplest local search. The average distance to the best solution is very high. The other local search variants have a lower distance to the best solution and a significantly higher FDC. In all cases the FDC is below or equal to 0.52, indicating that recombination may not be very helpful for these landscapes, since respectful recombination is known to be effective on correlated landscapes [22, 23]. Hence, the evolutionary local search algorithms based on mutation appear to be a good choice for effectively finding (near)optimum solutions to the MRT problem.

5 Conclusions

We have presented highly effective evolutionary algorithms incorporating local search for the minimum routing cost spanning tree problem arising in the design of P2P overlay topologies. The results show that the proposed algorithms are capable of approaching a lower bound on the problem effectively as well as improving upper bounds found by shortest path tree algorithms significantly. We proposed and investigated three types of local search in an evolutionary framework using a self-adapting mutation operator. The results show that either using a strong but time-consuming local search or using a scalable fast local search yields a solution quality below 0.5 % in short time.

There are several issues for future research. The influence of the adaptation rate used in our evolutionary framework has to be studied in detail. Moreover, it is important to investigate the scalability of the variants. Since there is no PlanetLab data with larger networks than those used here, we are forced to generate random instances with similar properties. Finally, we are working on distributed online algorithms for the MRT problem.

References

1. Gnutella: Gnutella Protocol v. 0.4 (2000) http://www.clip2.com/.
2. Ripeanu, M., Iamnitchi, A., Foster, I.: Mapping the Gnutella Network. IEEE Internet Computing (2002)
3. Cohen, B.: Incentives Build Robustness in BitTorrent. In: Workshop on Economics of Peer-to-Peer Systems, Berkeley, CA, USA (2003)
4. Chakravarti, A.J., Baumgartner, G., Lauria, M.: The Organic Grid: Self-Organizing Computation on a Peer-to-Peer Network. In: Proceedings of the International Conference on Autonomic Computing (ICAC '04), New York, NY (2004)
5. Merz, P., Gorunova, K.: Efficient Broadcast in P2P Grids. In: Proceedings of the IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid 2005), Cardiff, UK (2005)
6. Kubiatowicz, J., *et al.*: OceanStore: An Architecture for Global-Scale Persistent Storage. In: Proc. of the 9th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS 2000). (2000) 190–201

7. Demers, A.J., Greene, D.H., Hauser, C., Irish, W., Larson, J.: Epidemic algorithms for replicated database maintenance. In: Proceedings of the Sixth Annual ACM Symposium on Principles of Distributed Computing. (1987) 1–12
8. Johnson, D.S., Lenstra, J.K., Kan, A.H.G.R.: The Complexity of the Network Design Problem. Networks **8** (1978) 279–285
9. Wu, B.Y., Lancia, G., Bafna, V., Chao, K.M., Ravi, R., Tang, C.Y.: A Polynomial-Time Approximation Scheme for Minimum Routing Cost Spanning Trees. SIAM Journal on Computing **29** (1999) 761–778
10. Wu, B.Y., Chao, K.M., Tang, C.Y.: Approximation Algorithms for Some Optimum Communication Spanning Tree Problems. Discrete Applied Mathematics **102** (2000) 245–266
11. Dijkstra, E.W.: A Note on Two Problems in Connexion with Graphs. Numerische Mathematik **1** (1959) 269–271
12. Bellman, R.E.: On a Routing Problem. Quarterly of Applied Mathematics **16** (1958) 87–90
13. Ford, Jr., L.R., Fulkerson, D.R.: Flows in Networks. Princeton University Press (1962)
14. Sobeih, A., Wang, J., Yurcik, W.: Performance Evaluation and Comparison of Tree and Ring Application-Layer Multicast Overlay Networks. In: Proceedings of the 1st International Computer Engineering Conference: New Technologies for the Information Society (ICENCO), Cairo, Egypt (2004)
15. Lehmann, K.A., Kaufmann, M.: Evolutionary Algorithms for the Self-Organized Evolution of Networks. In et al., H.G.B., ed.: GECCO 2005: Proceedings of the 2005 Conference on Genetic and Evolutionary Computation. Volume 1., Washington DC, USA, ACM Press (2005) 563–570
16. Brosh, E., Shavitt, Y.: Approximation and Heuristic Algorithms for Minimum-Delay Application Layer Multicast Trees. In: The 23rd International Conference on Computer Copmunications, IEEE INFOCOM, Hong Kong (2004)
17. Tan, S.W., Waters, A., Crawford, J.: MeshTree: A Delay optimised Overlay Multicast Tree Building Protocol. Technical Report 5-05, University of Kent, Canterbury, UK (2005)
18. Lourenco, H.R., Martin, O., Stützle, T.: Iterated Local Search. In Glover, F., Kochenberger, G., eds.: Handbook of Metaheuristics. Kluwer Academic Publishers (2003)
19. Moscato, P.: On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts: Towards Memetic Algorithms. Technical Report C3P Report 826, Caltech Concurrent Computation Program, California Institue of Technology (1989)
20. Merz, P., Freisleben, B.: Memetic Algorithms for the Traveling Salesman Problem. Complex Systems **13** (2001) 297–345
21. Banerjee, S., Griffin, T., Pias, M.: The Interdomain Connectivity of PlanetLab Nodes. In Barakat, C., Pratt, I., eds.: Proceedings of the 5th International Workshop on Passive and Active Network Measurement, (PAM 2004). Volume 3015 of Lecture Notes in Computer Science., Springer (2004)
22. Merz, P., Freisleben, B.: Fitness Landscape Analysis and Memetic Algorithms for the Quadratic Assignment Problem. IEEE Transactions on Evolutionary Computation **4** (2000) 337–352
23. Merz, P.: Advanced Fitness Landscape Analysis and the Performance of Memetic Algorithms. Evolutionary Computation, Special Issue on Memetic Evolutionary Algorithms **12** (2004) 303–326