I N F O R M A T I K

Performance of heuristic and approximation algorithms for the uncapacitated facility location problem

Martin Hoefer

MPI–I–2002–1-005 December 2002

FORSCHUNGSBERICHT RESEARCH REPORT

M A X - P L A N C K - I N S T I T U T
F Ü R
I N F O R M A T I K

Stuhlsatzenhausweg 85 66123 Saarbrücken Germany

Authors' Addresses

Martin Hoefer

Technische Universität Clausthal
Marie-Hedwig-Straße 13
38678 Clausthal-Zellerfeld, Germany
email: Martin.Hoefer@TU-Clausthal.de

Max-Planck-Institut für Informatik
Stuhlsatzenhausweg 85
66123 Saarbrücken, Germany
email: mhoefer@mpi-sb.mpg.de

Abstract

The uncapacitated facility location problem (UFLP) is a problem that has been studied intensively in operational research. Recently a variety of new deterministic and heuristic approximation algorithms have evolved. In this paper, we compare five new approaches to this problem - the JMS- and the MYZ-approximation algorithms, a version of local search, a Tabu Search algorithm as well as a version of the Volume algorithm with randomized rounding. We compare solution quality and running times on different standard benchmark instances. With these instances and additional material a web page was set up [26], where the material used in this study is accessible.

Keywords

facility location; experimental study; benchmark library

1 Introduction

The problem of locating facilities and connecting clients at minimum cost has been one of the most studied problems in Operations Research. In this paper we focus on one of the simplest settings, the uncapacitated facility location problem (UFLP). The UFLP can be described as follows. We are given n possible facility locations and m cities. Let F denote the set of facilities and C the set of cities. Furthermore there are non-negative opening costs f_i for each facility $i \in F$ and connection costs c_{ij} for each connection between a facility i and a city j. The problem is to open a collection of facilities and connect each city to exactly one facility at minimum cost. The integer programming formulation of the UFLP is due to Balinski [5]:

$$
\begin{array}{lll}
\text{Min} & z = \sum_{ij} c_{ij} x_{ij} + \sum_i y_i f_i, & \\
& & \\
\text{subject to} & \sum_i x_{ij} = 1 & \text{for all } j \in C, \\
& y_i - x_{ij} \geq 0 & \text{for all } j \in C,\ i \in F, \\
& & \\
& x_{ij}, y_i \in \{0, 1\}. &
\end{array}
$$

We get the LP relaxation of this problem by setting $x_{ij}, y_i \in [0, 1]$. This LP relaxation is known to provide excellent lower bounds.
Instead of solving this problem to optimality, we will focus on finding approximate solutions. In the following we will present five methods, which are originating in different areas of optimization research. We will compare two approximation algorithms, two heuristics based on local search and one on LP-based approximation and rounding, which were recently developed and found to work good in practice.

2 Methods

2.1 Approximation algorithms

Recently a lot of approximation algorithms have evolved for the metric version of the UFLP in which the connection cost function c satisfies the triangular inequality. A couple of different techniques were used in these algorithms like LP-rounding ([12, 24]), greedy augmentation ([11]) or primal-dual methods ([20, 11]). In terms of computational hardness Guha and Khuller [14] showed that it is impossible to achieve an approximation guarantee of 1.463 unless $NP \in DTIME[n^{O(\log \log n)}]$. From the field of approximation algorithms we chose two of the newest and most promising variants.

2.1.1 JMS-Algorithm

The JMS-Algorithm uses a greedy method to improve the solution. The notion of time that is involved was introduced in an earlier 3-approximation algorithm by Jain and Vazirani [20]. Later on Mahdian et al. [21] translated the primal-dual scheme into a greedy 1.861-approximation algorithm. In the third paper Jain, Mahdian and Sabieri [19] presented the JMS-Algorithm (JMS), which improved the approximation bound to 1.61. However, it had a slightly worse complexity of $O(n^3)$ instead of $O(n^2\ log\ n)$.
The following sketch of JMS is taken from [22]:

1. At first all cities are unconnected, all facilities unopened, and the budget of every city j, denoted by B_j, is initialized to 0. At every moment, each city j offers some money from its budget to each unopened facility i. The amount of this offer is equal to $max(B_j - c_{ij}, 0)$ if j is unconnected, and $max(c_{i'j} - c_{ij}, 0)$ if it is connected to some other facility i'.

2. While there is an unconnected city, increase the budget of each unconnected city at the same rate, until one of the following events occurs:

   (a) For some unopened facility i, the total offer that it receives from cities is equal to the cost of opening i. In this case, we open facility i, and for every city j (connected or unconnected) which has a non-zero offer to i, we connect j to i.

   (b) For some unconnected city j, and some facility i that is already open, the budget of j is equal to the connection cost c_{ij}. In this case, we connect j to i.

One important property of the solution of this algorithm is that it cannot be improved by simply opening an unopened facility. This is the main advantage over the previous 1.861-algorithm in [21]. In [19] experiments revealed an appealing behavior of JMS in practice.

2.1.2 MYZ Algorithm

The MYZ algorithm could further improve the approximation factor of JMS. Mahdian, Ye and Zhang [22] applied scaling and greedy augmentation to the algorithm. For the resulting MYZ Algorithm (MYZ) the authors could prove an approximation factor of 1.52 for the metric UFLP, which is the best known so far for this problem for any algorithm.
MYZ is outlined below. In step 4 of the algorithm C is the total connection cost of the present solution and C' the connection cost after opening a facility u.

1. Scale up all opening costs by a factor of $\delta = 1.504$
2. Solve the scaled instance with JMS
3. Scale down all opening costs by the same factor δ
4. **while** there is a unopened facility u, for which the ratio $(C - C' - f_u)/f_u$ is maximized and positive, open facility u and update solution

2.2 Heuristic and randomized algorithms

In terms of meta-heuristics there has not been such an intense research activity. A simulated annealing algorithm [3] was developed, which produces good results to the expense of high computation costs. Tabu search algorithms have been very successful in solving the UFLP (see [2, 23, 25]). A very elaborate genetic algorithm has been proposed by Kratica et al. over a series of papers ([16, 17, 18]). Their final version involves clever implementation techniques and finds optimal solutions for all the examined benchmarks.

2.2.1 Tabu Search

In [23] Van Hentenryck and Michel proposed a simple tabu search algorithm that works very fast and outperforms the genetic algorithm in [18] in terms of solution quality, robustness and execution time. Therefore we used this algorithm in our comparative study.
The tabu search algorithm uses a slightly different representation of the problem. For a solution of the UFLP it is enough to know the set $S \subseteq F$ of opened facilities. Cities are connected to the cheapest opened facility, i.e. city j is connected to $i \in S$ with $c_{ij} = \min_{i' \in S} c_{i'j}$. A neighborhood move from S to S' is defined as flipping the status of a facility from opened to closed ($S' = S \backslash i$) or vice versa ($S' = S \cup i$). When the status of a facility was flipped, flipping back this facility becomes prohibited (i.e. tabu) for a number of iterations. The number of iterations is adjusted using a standard scheme (see [23] for details). The high level algorithm can be stated as follows:

1. $S \leftarrow$ an arbitrary feasible solution
2. Set cost$(S^*) = \infty$
3. **do**
4. $bestgain$ = maximum cost savings over all possible non-tabu flips
5.   **if** ($bestgain > 0$)

6. Apply random flip with best gain, update tabu lists and list length

7. **else** close random facility

8. Update S - connections of cities and datastructures

9. **if** (cost(S) < cost(S^*)) do $S^* \leftarrow S$

10. **while** change of S^* in the last 500 iterations

11. return S^*

For every city j the algorithm uses three pieces of information: The number of the opened facility with the cheapest connection to j, the cost of this connection and the cost of the second cheapest connection to an opened facility. With this information the gains of opening and closing a facility can be updated incrementally in step 8. Thereby a direct evaluation of the objective function can be avoided. The algorithm uses priority queues to determine the second cheapest connections for each city. Due to these techniques the algorithm has a running time of $O(m \log n)$ in each iteration.

2.2.2 Local Search

The local search community has only paid limited attention to the UFLP so far. Apart from the tabu search algorithms there have been a few simple local search procedures proposed in [15, 11]. In this paper we use the simple version of Arya et al [4], which can be stated as follows:

1. $S \leftarrow$ an arbitrary feasible solution

2. **while** there is an operation `op` such that
   cost of $\mathtt{op}(S) \leq (1 - \frac{\epsilon}{p(n,m)})\text{cost}(S)$
   do $S \leftarrow \mathtt{op}(S)$

3. return S

The solution S again is the set of opened facilities. The operation `op` is defined as opening or closing a facility or exchanging the status of an opened and a closed facility. The parameters were set to $\epsilon = 0.1$ and $p(n, m) = n + m$. For this algorithm the authors could prove an approximation guarantee of 3 on metric instances.
To improve the running time of the algorithm we incorporated the use of incremental datastructures from the Tabu search algorithm and preferences for the simple moves as follows. We generally prefer applying the simple flips of opening and closing a facility (denoted as `ops`). As in the Tabu Search we apply one random flip of the

flips resulting in best gain of the cost function. When these flips do not satisfy the acceptance condition, we pick an exchange move that would give enough improvement. If there is no such move left the algorithm stops. The modified version can be stated as follows:

1. $S \leftarrow$ an arbitrary feasible solution
2. $exitloop \leftarrow false$
3. **while** $exitloop = false$
4. **while** there is an `ops` such that $\text{cost}(\texttt{ops}(S)) \leq (1 - \frac{\epsilon}{p(n,m)})\ \text{cost}(S)$
5. find a random $\texttt{ops}^*$ of the `ops` with best gain
6. do $S \leftarrow \texttt{ops}^*(S)$
7. **if** there is an `op` such that $\text{cost}(\texttt{op}(S)) \leq (1 - \frac{\epsilon}{p(n,m)})\ \text{cost}(S)$
8. do $S \leftarrow \texttt{op}(S)$
9. **else** $exitloop \leftarrow true$
10. return S

Arya et al. suggested that the algorithm should be combined with the standard scaling techniques [11] to improve the approximation factor to 2.414. Interestingly this version performs inferior in practice. Therefore the version without scaling (denoted as LOCAL) was used for the comparison with the other algorithms. A comparison between the unscaled and scaled versions can be found in section 3.5.

2.2.3 Volume algorithm

For some of the test instances we obtained a lower bound using a version of the Volume algorithm, which was developed by Barahona in [7]. The Volume algorithm is an iterated subgradient optimization procedure, which is able to provide a primal solution and a lower bound on the optimal solution cost. To improve solution quality and speed up the computation Barahona and Chudak [8] used the rounding heuristic (RRWC) presented in [12] to find good upper bounds on the optimal dual solution cost and therefore reduce the iterations of the Volume algorithm. However, this approach has generally very high execution times in comparison to the other methods presented here. Instead we used a faster version of this algorithm which uses only a basic randomized rounding procedure and slightly different parameter settings. It will be denoted by V&RR and is available on the web page of the COIN-OR project by IBM [6]. Regarding solution quality and running time this algorithm is generally inferior to the other algorithms. The results should only be seen as benchmark values of available optimization code. We will not go into detail describing the method, the code or the parameter settings here. The interested reader is referred to [6, 7, 8] for the specific details of the algorithm and the implementation.

3 Experiments

We tested all given algorithms on several sets of benchmark instances. The instances were chosen to cover different types of facility location problems. First we studied the Bilde-Krarup benchmarks, which were proposed in [10]. These are non-metric small scale instances with $n \times m = 30 \times 80$ - 50×100. Next we focused on small scale benchmarks proposed by Galvão and Raggi in [13]. These are metric instances with $n = m = 50$ - 200. Then we examined the performance on the `cap` instances from the ORLIB [9] and the M* instances, which were proposed in [18]. These are non-metric small and medium sized instances with $n \times m = 16 \times 50$ - 2000×2000. Finally we studied metric large scale instances with $n = m = 500 - 3000$, which were proposed in [1] and used as UFLP benchmarks in [7]. On all instances we averaged over the performance of 20 runs for each algorithm. The experiments were done on a 866Mhz Intel Pentium III running Linux. For most problems we used CPLEX to solve the problems to optimality. The CPLEX-runs were done on a 333Mhz Sun Enterprise 10000 with UltraSPARC processors running UNIX. The execution times are about a factor of 2.5 times higher than the times for the algorithms.
With all benchmark instances, implementations of all algorithms and benchmark generators a web page was set up. All material used in this study can be accessed online at the UflLib [26].

3.1 Bilde-Krarup Instances

The Bilde-Krarup instances are small scale instances of 22 different types. The costs for the different types are calculated with the parameters given in Table 1. As the exact instances are not known, we generated 10 test instances for each problem type. In Table 2 we report the results for the deterministic algorithms and in Table 3 the results for the heuristic and randomized algorithms. In columns 'Opt' we report the number of instances that could be solved to optimality. For the heuristic algorithms we also report the average percentage of runs on the instances solved to optimality that ended with an optimal solution. In columns 'Error' we report the average error of the final solution, in columns 'Time' the average execution time in seconds. In column 'CPX' we denoted the average running time of CPLEX to solve the instances.

The deterministic algorithms perform quite good in these instances. The average error is 2.607% at maximum although the problems are not of metric nature. MYZ

Type	Size	f_i	c_{ij}
B	50x100	Discrete Uniform (1000, 10000)	Discrete Uniform (0,1000)
C	50x100	Discrete Uniform (1000, 2000)	Discrete Uniform (0,1000)
Dq*	30x80	Identical, 1000*q	Discrete Uniform (0,1000)
Eq*	50x100	Identical, 1000*q	Discrete Uniform (0,1000)

* q=1,...,10

Table 1: Parameters for the Bilde-Krarup problem classes

Type	CPX	JMS			MYZ		
		Opt	Error	Time	Opt	Error	Time
B	6.859	5	0.416%	0.003	4	0.588%	0.003
C	107.558	1	1.750%	0.003	3	0.886%	0.003
D1	21.591	0	2.445%	0.001	1	1.689%	0.002
D2	30.990	1	1.675%	0.002	2	1.133%	0.002
D3	28.103	1	2.607%	0.002	4	0.923%	0.002
D4	26.685	3	0.796%	0.002	3	0.597%	0.002
D5	22.368	4	0.647%	0.002	7	0.085%	0.002
D6	28.393	2	1.042%	0.002	3	1.315%	0.002
D7	24.484	1	1.771%	0.002	6	0.664%	0.002
D8	20.947	4	1.587%	0.002	4	1.044%	0.002
D9	22.326	7	0.846%	0.002	9	0.012%	0.002
D10	19.122	7	0.252%	0.002	8	0.189%	0.002
E1	133.839	2	2.265%	0.003	3	1.317%	0.003
E2	229.305	2	1.650%	0.003	4	0.845%	0.003
E3	190.860	2	1.610%	0.003	2	0.940%	0.003
E4	185.168	3	1.192%	0.003	3	0.781%	0.004
E5	163.571	1	2.560%	0.003	7	0.690%	0.004
E6	173.918	4	1.049%	0.003	5	0.661%	0.004
E7	164.845	5	0.759%	0.004	5	0.613%	0.004
E8	180.186	1	1.474%	0.004	4	0.887%	0.004
E9	174.150	3	1.232%	0.004	6	0.674%	0.004
E10	148.229	4	0.775%	0.004	6	0.404%	0.004

Table 2: Results for the deterministic algorithms

Type	LOCAL			TABU-Search			V&RR		
	Opt	Error	Time	Opt	Error	Time	Opt	Error	Time
B	8, [100]	0.046%	0.012	10, [100]	0.000%	0.053	10, [69.5]	0.419%	0.421
C	5, [84.0]	0.848%	0.014	7, [91.43]	0.245%	0.055	1, [5.0]	4.454%	0.525
D1	3,[81.7]	1.678%	0.006	6,[90.8]	0.241%	0.038	1,[15.0]	3.719%	0.239
D2	2,[100]	1.758%	0.006	9,[73.9]	0.537%	0.044	2,[22.5]	3.083%	0.254
D3	3,[96.7]	0.879%	0.006	9,[99.4]	0.073%	0.042	3,[30.0]	2.245%	0.235
D4	8,[89.4]	0.530%	0.006	10,[100]	0.000%	0.041	9,[36.1]	1.248%	0.243
D5	6,[97.5]	0.402%	0.006	10,[94.0]	0.004%	0.040	8,[47.5]	0.995%	0.246
D6	5,[100]	0.882%	0.006	9,[96.7]	0.146%	0.042	7,[60.0]	0.919%	0.259
D7	8,[100]	0.354%	0.005	10,[100]	0.000%	0.042	10,[79.5]	0.214%	0.251
D8	7,[90.0]	1.000%	0.006	9,[100]	0.166%	0.043	8,[50.6]	1.390%	0.259
D9	8,[100]	0.285%	0.006	10,[100]	0.000%	0.043	10,[73.0]	0.496%	0.256
D10	7,[90.7]	0.760%	0.006	10,[92.5]	0.139%	0.043	10,[74.0]	0.506%	0.268
E1	1,[100]	1.430%	0.013	10,[57.0]	0.388%	0.062	0,[0.0]	5.712%	0.516
E2	3,[45.0]	2.712%	0.013	10,[93.5]	0.006%	0.067	0,[0.0]	4.479%	0.560
E3	4,[93.8]	0.784%	0.012	7,[89.3]	0.268%	0.061	5,[10.0]	3.419%	0.553
E4	3,[80.0]	1.577%	0.013	9,[100]	0.013%	0.060	4,[11.3]	2.505%	0.581
E5	4,[73.8]	2.019%	0.013	10,[100]	0.000%	0.062	9,[23.3]	1.924%	0.546
E6	7,[79.3]	0.969%	0.013	10,[100]	0.000%	0.062	10,[22.0]	1.981%	0.602
E7	6,[86.7]	0.996%	0.015	10,[100]	0.000%	0.063	6,[19.2]	1.802%	0.586
E8	4,[100]	1.043%	0.014	10,[89.5]	0.177%	0.067	6,[61.7]	1.318%	0.585
E9	7,[92.9]	0.655%	0.013	10,[100]	0.000%	0.066	9,[46.7]	0.896%	0.592
E10	8,[92.5]	0.948%	0.013	10,[100]	0.000%	0.066	10,[50.0]	0.864%	0.598

Table 3: Results for the heuristic and randomized algorithms

Size	δ	Parameters for f_i	
		mean	stand. dev.
50	0.061	25.1	14.1
70	0.043	42.3	20.7
100	0.025	51.7	28.9
150	0.018	186.1	101.5
200	0.015	149.5	94.4

Table 4: Parameters for the Galvão-Raggi problem classes

performs significantly better than JMS in terms of solution quality. It can solve 37 more problems to optimality and also has a lower average error. The execution time is slightly higher because it uses JMS as a subroutine.
For the heuristic algorithms TABU provides the best results. It was able to solve problems of all classes to optimality in a high number of runs. Unfortunately it also is much slower than LOCAL, MYZ and JMS.
LOCAL also performs competitive on most of these problem classes. Compared to TABU it is able to solve problems of all classes to optimality, but the overall number of instances solved is very much lower. In terms of the running time it is much faster though.
V&RR is outperformed by any of the other algorithms. It reveals the highest running time and the worst solution quality.

3.2 Galvão-Raggi Instances

Galvão and Raggi proposed unique benchmarks for the UFLP. A graph is given with an arc density δ, which is defined as $\delta =$ connections present $/(m * n)$. Each present connection has a cost sampled from a uniform distribution in the range $[1, n]$ (except for $n = 150$, where the range is $[1, 500]$). The connection costs between a facility i and a city j are determined by the shortest path from i to j in the given graph. The opening costs f_i are assumed to come from a Normal distribution. Originally Galvão and Raggi proposed problems with $n = m = 10, 20, 30, 50, 70, 100, 150$ and 200. We will consider the 5 largest types. The density values and the parameters for the Normal distribution are listed in Table 4. The exact instances for these benchmarks are not known. So as for the Bilde-Krarup benchmarks we generated 10 instances for each class. The results for the deterministic algorithms are reported in Table 5 and for the randomized and heuristic algorithms in Table 6. In columns 'Opt' the number of instances solved to optimality is reported. For the instances solved to optimality by a specific algorithm we averaged the percentage of runs that ended with the optimal solution and report this number in brackets. In columns 'Error' we

Type	CPX	JMS			MYZ		
		Opt	Error	Time	Opt	Error	Time
50	0.200	10	0.000%	0.001	10	0.032%	0.001
70	0.332	9	0.038%	0.003	7	0.065%	0.003
100	0.677	9	0.014%	0.006	8	0.099%	0.007
150	1.623	7	0.059%	0.016	6	0.111%	0.016
200	3.355	6	0.071%	0.036	7	0.032%	0.036

Table 5: Results for the deterministic algorithms

Type	LOCAL			TABU			V&RR		
	Opt	Error	Time	Opt	Error	Time	Opt	Error	Time
50	9,[99.5]	0.236%	0.006	10,[100]	0.000%	0.026	10,[97.0]	0.007%	0.112
70	7,[80.0]	0.063%	0.013	9,[100]	0.061%	0.037	10,[93.0]	0.001%	0.238
100	4,[86.3]	0.022%	0.026	10,[83.5]	0.039%	0.055	10,[90.0]	0.002%	0.965
150	5,[92.0]	0.020%	0.062	9,[55.6]	0.239%	0.085	9, [94.4]	0.001%	3.375
200	6,[65.8]	0.022%	0.127	9,[53.9]	0.131%	0.133	10,[68.0]	0.011%	7.363

Table 6: Results for the heuristic algorithms

report the average error, in columns 'Time' the average execution time in seconds. We also included the average running times of CPLEX in column 'CPX' of Table 5.

JMS performs on these metric instances slightly better than MYZ. For the heuristic and randomized algorithms V&RR performs very good - even better than TABU - to the expense of high execution times. In fact, the times are prohibitively high as the algorithm needs much more time than CPLEX to solve the instances to optimality.

3.3 ORLIB and M* Instances

The `cap` problems from the ORLIB are non-metric medium sized instances. The M* instances were designed to represent classes of real UFLPs. They are very challenging for mathematical programming methods because they have a large number of suboptimal solutions. In Table 7 we report the results for the deterministic and in Tables 8 and 9 the results for the heuristic and randomized algorithms. For the deterministic algorithms we indicate with a star in columns 'Opt', whether an instance was solved to optimality or not. For the heuristic and randomized algorithms columns 'Opt' show the percentage of runs in which the algorithm was able to solve the problem to optimality. In Columns 'Cost' and 'Error' we report the average cost and the error of the final solution over all runs. For the heuristic and randomized al-

gorithms we report the average solution ('Avg') and the standard deviation ('$\sigma(S)$') expressed as a percentage of the average solution. We separately report the best solution ('Best') if it is not equal to the optimum or the average solution. As TABU always managed to find optimal solutions, we omit the column 'Best' here.
For the larger benchmarks the optimal solutions are not known. Instead we used the best solutions found as a reference, which for all benchmarks were encountered by TABU. All values that do not relate to an optimal solution are denoted in brackets.
Running times for all algorithms can be found in Table 10. In Columns 'Time' we report the average running time in seconds. For the heuristic and randomized algorithms we also included the standard deviation ('Std') for the running time expressed as a percentage of the average running time.
At the end of each table we summarized the results for the groups with more than one instance. We put the data in the format of the previous tables for the Galvão-Raggi and Bilde-Krarup instances.

Bench	Size	Opt Cost	JMS			MYZ		
			Opt	Cost	Error	Opt	Cost	Error
cap71	16x50	932615.75		932615.75	0.000%	*	932615.75	0.000%
cap72	16x50	977799.40		981538.85	0.382%		981538.85	0.382%
cap73	16x50	1010641.45		1015508.94	0.482%		1012643.69	0.198%
cap74	16x50	1034976.98		1042643.69	0.741%		1045383.79	1.006%
cap101	25x50	796648.44		798591.13	0.244%		797508.73	0.108%
cap102	25x50	854704.20		858109.33	0.398%	*	854704.20	0.000%
cap103	25x50	893782.11		902413.26	0.966%		895027.19	0.139%
cap104	25x50	928941.75		932527.19	0.386%		932007.96	0.330%
cap131	50x50	793439.56		795382.25	0.245%		794299.85	0.108%
cap132	50x50	851495.33		854900.45	0.400%	*	851495.33	0.000%
cap133	50x50	893076.71		901481.84	0.941%		894095.76	0.114%
cap134	50x50	928941.75		932527.19	0.386%		932007.96	0.330%
capa	100x1000	17156454.48		17765201.95	3.548%		17902353.24	4.348%
capb	100x1000	12979071.58		13070745.09	0.706%		13271844.16	2.256%
capc	100x1000	11505594.33		11702914.76	1.715%		11681971.18	1.533%
MO1	100x100	1305.95	*	1305.95	0.000%	*	1305.95	0.000%
MO2	100x100	1432.36		1479.11	3.264%		1460.29	1.950%
MO3	100x100	1516.77		1521.47	0.310%		1521.47	0.310%
MO4	100x100	1442.24	*	1442.24	0.000%	*	1442.24	0.000%
MO5	100x100	1408.77		1413.81	0.358%	*	1408.77	0.000%
MP1	200x200	2686.48	*	2686.48	0.000%	*	2686.48	0.000%
MP2	200x200	2904.86		2914.42	0.329%		2914.42	0.329%
MP3	200x200	2623.71		2658.98	1.345%	*	2623.71	0.000%
MP4	200x200	2938.75	*	2938.75	0.000%	*	2938.75	0.000%
MP5	200x200	2932.33		2939.95	0.260%		2939.95	0.260%
MQ1	300x300	4091.01	*	4091.01	0.000%	*	4091.01	0.000%
MQ2	300x300	4028.33	*	4028.33	0.000%	*	4028.33	0.000%
MQ3	300x300	4275.43		4307.97	0.761%	*	4275.43	0.000%
MQ4	300x300	4235.15		4273.05	0.895%		4239.23	0.096%
MQ5	300x300	4080.74		4103.75	0.564%		4103.75	0.564%
MR1	500x500	[2608.15]		2614.72	[0.252%]		2609.13	[0.038%]
MR2	500x500	[2654.74]	[*]	2654.74	[0.000%]	[*]	2654.74	[0.000%]
MR3	500x500	[2788.25]		2794.41	[0.221%]		2794.41	[0.221%]
MR4	500x500	[2756.04]		2782.28	[0.952%]		2773.89	[0.648%]
MR5	500x500	[2505.05]		2517.10	[0.481%]		2529.87	[0.991%]
MS1	1000x1000	[5283.76]	[*]	5283.76	[0.000%]	[*]	5283.76	[0.000%]
MT1	2000x2000	[10069.80]		10090.49	[0.205%]		10090.49	[0.205%]
cap7*			0		0.401%	1		0.397%
cap10*			0		0.499%	1		0.144%
cap13*			0		0.493%	1		0.138%
capa-c			0		1.990%	0		2.712%
MO*			2		0.786%	3		0.452%
MP*			2		0.387%	3		0.118%
MQ*			2		0.444%	3		0.132%
MR*			[1]		[0.381%]	[1]		[0.380%]

Table 7: Solution quality of the deterministic algorithms

Bench	LOCAL					TABU				V&RR			
	Opt	Best	Avg	Error	Std	Opt	Avg	Error	Std	Opt	Avg	Error	Std
cap71	100%		932615.75	0.000%	0.000%	100%	932615.75	0.000%	0.000%	100%	932615.75	0.000%	0.000%
cap72	0%		979099.61	0.133%	0.000%	100%	977799.40	0.000%	0.000%	100%	977799.40	0.000%	0.000%
cap73	0%		1011067.65	0.042%	0.000%	100%	1010641.45	0.000%	0.000%	100%	1010641.45	0.000%	0.000%
cap74	100%		1034976.98	0.000%	0.000%	100%	1034976.98	0.000%	0.000%	100%	1034976.98	0.000%	0.000%
cap101	0%		797582.29	0.117%	0.000%	100%	796648.44	0.000%	0.000%	100%	796648.44	0.000%	0.000%
cap102	100%		854704.20	0.000%	0.000%	100%	854704.20	0.000%	0.000%	100%	854704.20	0.000%	0.000%
cap103	100%		893782.11	0.000%	0.000%	100%	893782.11	0.000%	0.000%	100%	893782.11	0.000%	0.000%
cap104	0%		930026.55	0.117%	0.000%	100%	928941.75	0.000%	0.000%	100%	928941.75	0.000%	0.000%
cap131	100%		793439.56	0.000%	0.000%	100%	793439.56	0.000%	0.000%	100%	793439.56	0.000%	0.000%
cap132	100%		851495.33	0.000%	0.000%	100%	851495.33	0.000%	0.000%	100%	851495.33	0.000%	0.000%
cap133	0%		895292.08	0.248%	0.000%	100%	893076.71	0.000%	0.000%	40%	893688.55	0.069%	0.057%
cap134	0%		935422.70	0.698%	0.000%	100%	928941.75	0.000%	0.000%	100%	928941.75	0.000%	0.000%
capa	100%		17156454.48	0.000%	0.000%	100%	17156454.48	0.000%	0.000%	100%	17156454.48	0.000%	0.000%
capb	50%		13041143.92	0.478%	0.575%	75%	13000649.83	0.166%	0.289%	100%	12979071.58	0.000%	0.000%
capc	0%	11509361.7	11534161.39	0.248%	0.084%	70%	11513112.75	0.065%	0.117%	10%	11519212.05	0.100%	0.130%
cap7*	2,[100]			0.044%		4,[100]		0.000%		4,[100]		0.000%	
cap10*	2,[100]			0.059%		4,[100]		0.000%		4,[100]		0.000%	
cap13*	2,[100]			0.236%		4,[100]		0.000%		4,[85.0]		0.017%	
capa-c	2,[75.0]			0.242%		3,[81.7]		0.077%		3,[70.0]		0.039%	

Table 8: Solution quality of the heuristic and randomized algorithms

Bench	LOCAL					TABU				V&RR				
	Opt	Best	Avg	Error	Std	Opt	Avg	Error	Std	Opt	Best	Avg	Error	Std
MO1	100%		1305.95	0.000%	0.000%	100%	1305.95	0.000%	0.000%	10%		1315.833	0.757	0.437%
MO2	15%		1450.81	1.288%	0.534%	100%	1432.36	0.000%	0.000%	15%		1449.311	1.184	0.564%
MO3	0%		1521.47	0.310%	0.000%	100%	1516.77	0.000%	0.000%	5%		1536.238	1.283	0.584%
MO4	100%		1442.24	0.000%	0.000%	100%	1442.24	0.000%	0.000%	20%		1465.982	1.646	1.016%
MO5	100%		1408.77	0.000%	0.000%	100%	1408.77	0.000%	0.000%	65%		1410.879	0.150	0.239%
MP1	100%		2686.48	0.000%	0.000%	100%	2686.48	0.000%	0.000%	15%		2709.68	0.864%	0.535%
MP2	100%		2904.86	0.000%	0.000%	100%	2904.86	0.000%	0.000%	5%		2944.39	1.361%	0.672%
MP3	100%		2623.71	0.000%	0.000%	100%	2623.71	0.000%	0.000%	10%		2666.66	1.637%	0.618%
MP4	0%	2942.63	2943.99	0.178%	0.051%	100%	2938.75	0.000%	0.000%	0%	2944.78	2983.59	1.526%	0.680%
MP5	40%		2939.74	0.252%	0.251%	100%	2932.33	0.000%	0.000%	0%	2939.95	2950.67	0.625%	0.195%
MQ1	100%		4091.01	0.000%	0.000%	100%	4091.01	0.000%	0.000%	15%		4151.64	1.482%	0.834%
MQ2	100%		4028.33	0.000%	0.000%	100%	4028.33	0.000%	0.000%	5%		4103.26	1.860%	0.658%
MQ3	100%		4275.43	0.000%	0.000%	100%	4275.43	0.000%	0.000%	10%		4326.12	1.186%	0.694%
MQ4	100%		4235.15	0.000%	0.000%	100%	4235.15	0.000%	0.000%	5%		4291.00	1.319%	0.651%
MQ5	85%		4084.73	0.098%	0.235%	100%	4080.74	0.000%	0.000%	0%	4127.22	4154.24	1.801%	0.340%
MQ1	100%		4091.01	0.000%	0.000%	100%	4091.01	0.000%	0.000%	0%	4114.94	4161.48	1.722%	0.564%
MQ2	100%		4028.33	0.000%	0.000%	100%	4028.33	0.000%	0.000%	5%		4096.23	1.686%	0.679%
MQ3	100%		4275.43	0.000%	0.000%	100%	4275.43	0.000%	0.000%	10%		4305.77	0.710%	0.434%
MQ4	100%		4235.15	0.000%	0.000%	100%	4235.15	0.000%	0.000%	0%	4239.24	4284.37	1.162%	0.599%
MQ5	65%		4089.08	0.204%	0.278%	100%	4080.74	0.000%	0.000%	5%		4144.23	1.556%	0.607%
MR1	[20%]		2612.76	[0.177%]	0.317%	[100%]	2608.15	[0.000%]	0.000%	[0%]	2614.70	2633.31	[0.965%]	0.434%
MR2	[45%]		2679.61	[0.936%]	0.864%	[100%]	2654.74	[0.000%]	0.000%	[0%]	2697.65	2729.38	[2.812%]	0.798%
MR3	[80%]		2789.17	[0.033%]	0.066%	[100%]	2788.25	[0.000%]	0.000%	[0%]	2793.32	2838.13	[1.789%]	0.707%
MR4	[100%]		2756.04	[0.000%]	0.000%	[100%]	2756.04	[0.000%]	0.000%	[0%]	2784.47	2821.06	[2.359%]	0.658%
MR5	[100%]		2505.05	[0.000%]	0.000%	[100%]	2505.05	[0.000%]	0.000%	[0%]	2532.28	2559.81	[2.186%]	0.614%
MS1	[100%]		5283.76	[0.000%]	0.000%	[100%]	5283.76	[0.000%]	0.000%	[0%]	5327.17	5380.411	[1.829%]	0.572%
MT1	[20%]		10085.84	[0.159%]	0.080%	[90%]	10071.77	[0.020%]	0.059% [0%]	10121.95	10252.72	[1.817%]	0.462%	
MO*	4,[78.8]			0.320%		5,[100]		0.000%		5,[23.0]			1.004%	
MP*	4,[85.0]			0.086%		5,[100]		0.000%		3,[10.0]			1.203%	
MQ*	5,[93.0]			0.041%		5,[100]		0.000%		3,[6.7]			1.367%	
MR*	5,[69.0]			[0.229%]		5,[100]		[0.000%]		0,[0.0]			[2.022%]	

Table 9: Solution quality of the heuristic and randomized algorithms

Again the deterministic algorithms perform very well on the benchmarks. The maximum error for both methods was produced on the `capa` benchmark. Of the deterministic algorithms MYZ did perform better than JMS. It was able to solve additional 6 problems to optimality. JMS could only achieve a better performance in 4 of the 37 benchmarks. In terms of running time MYZ becomes slightly less competitive on larger problems because the additional calculations of the greedy augmentation procedure need more time.
With a maximum average error of 0.289% TABU is again the algorithm with the best performance on these benchmarks. It is able to solve all problems to optimality - in most cases with a high frequency. Here our results are consistent with the values reported in [23]. However, the running times of our code are significantly faster than the times needed by the implementation of Michel and Van Hentenryck (a factor of 2 and more).
Compared to TABU the solution quality of LOCAL is not very competitive. It fails to find optimal solutions on 9 problems, while 7 of them are `cap`-benchmarks. The running times, however, are very competitive, as it performs in most cases significantly better than TABU.
The performance of V&RR is not very good in comparison to the other methods. On some of the `cap` instances the algorithm achieves good solution quality. On the M*-instances, however, it performs worse than all other algorithms in terms of solution quality and execution time. The execution times for the small problems exceed the times of CPLEX again. The practical use of this algorithm for smaller problems should therefore be avoided. For problems with $m, n \geq 100$, however, execution times of CPLEX become significantly higher.
Interestingly there is hardly any variation of the running times of V&RR on the M*-instances.

3.4 k-median Instances

In this section we take a look at large scale instances for the UFLP. The benchmarks considered here were originally introduced for the k-median problem in [1]. In [8] they were used as test instances for the UFLP. To construct an instance, we pick n points independent uniformly at random in the unit square. Each point is simultaneously city and facility. The connection costs are the Euklidian distances in the plane. All facility opening costs are identical. To prevent numerical problems and preserve the metric properties, we rounded up all data to 4 significant digits and then made all the data entries integer.
In [1] the authors showed that, when n is large, any enumerative method based on the lower bound of the relaxed LP would need to explore an exponential number of solutions. They also showed that the solution of the relaxed LP is, asymptotically in the number of points, about 0.998% of the optimum.

Bench	CPX	JMS	MYZ	LOCAL		TABU		V&RR	
		Time	Time	Time	Std	Time	Std	Time	Std
cap71	0.109	0.001	0.001	0.002	1.176%	0.022	0.114%	0.060	3.275%
cap72	0.067	0.001	0.001	0.002	0.420%	0.022	1.103%	0.051	7.031%
cap73	0.078	0.001	0.001	0.002	0.516%	0.023	0.075%	0.043	0.094%
cap74	0.068	0.001	0.001	0.002	0.379%	0.024	0.277%	0.033	4.221%
cap101	0.106	0.001	0.001	0.003	0.456%	0.024	0.096%	0.086	4.019%
cap102	0.093	0.001	0.001	0.003	0.437%	0.023	0.060%	0.078	2.091%
cap103	0.094	0.001	0.001	0.005	0.294%	0.026	24.885%	0.064	6.961%
cap104	0.098	0.001	0.001	0.003	4.750%	0.024	0.071%	0.063	8.096%
cap131	0.206	0.002	0.002	0.006	0.372%	0.026	0.094%	0.200	5.441%
cap132	0.186	0.002	0.002	0.007	0.329%	0.025	0.086%	0.148	4.138%
cap133	0.192	0.002	0.002	0.005	0.387%	0.027	7.594%	0.146	11.840%
cap134	0.199	0.002	0.002	0.006	0.492%	0.030	2.150%	0.141	20.066%
capa	48.834	0.153	0.162	0.404	17.640%	1.343	4.517%	15.371	11.267%
capb	37.746	0.151	0.158	0.545	21.985%	0.948	11.628%	20.505	7.117%
capc	146.654	0.152	0.155	0.480	27.173%	0.924	16.881%	22.495	12.788%
MO1	165.811	0.008	0.008	0.025	7.546%	0.060	0.317%	1.940	3.616%
MO2	154.922	0.008	0.008	0.024	9.879%	0.063	3.647%	2.077	2.925%
MO3	201.240	0.008	0.008	0.028	5.489%	0.082	17.751%	1.663	2.281%
MO4	80.766	0.008	0.008	0.035	0.335%	0.060	1.178%	1.516	4.701%
MO5	115.189	0.008	0.008	0.024	0.262%	0.069	3.456%	2.012	3.111%
MP1	4442.243	0.049	0.050	0.180	10.829%	0.228	3.055%	9.812	0.458%
MP2	9307.855	0.050	0.051	0.169	31.750%	0.225	2.385%	9.826	0.350%
MP3	1183.319	0.049	0.051	0.196	23.053%	0.224	1.966%	11.573	1.618%
MP4	11219.924	0.049	0.050	0.154	19.691%	0.267	17.672%	12.210	2.110%
MP5	13288.276	0.049	0.051	0.129	7.645%	0.230	6.578%	12.725	1.194%
MQ1	25876.314	0.141	0.143	0.563	19.773%	0.668	1.613%	24.667	2.603%
MQ2	44236.625	0.141	0.142	0.507	20.969%	0.646	2.563%	26.538	1.597%
MQ3	21227.507	0.146	0.144	0.603	16.013%	0.683	1.499%	27.111	0.356%
MQ4	28484.952	0.143	0.145	0.471	11.159%	0.671	2.145%	20.732	0.495%
MQ5	126890.793	0.139	0.141	0.472	21.826%	0.675	4.890%	26.088	0.473%
MR1		0.468	0.481	2.388	33.574%	1.796	11.183%	78.575	0.902%
MR2		0.455	0.472	1.879	18.124%	1.958	7.453%	78.762	0.500%
MR3		0.464	0.480	2.112	8.349%	2.051	15.839%	85.650	0.476%
MR4		0.471	0.477	1.961	16.183%	1.656	1.825%	77.825	0.417%
MR5		0.461	0.471	2.339	21.605%	1.664	3.037%	76.141	0.478%
MS1		2.281	2.323	11.720	14.147%	6.366	1.705%	304.066	0.978%
MT1		11.079	11.241	89.592	22.647%	31.505	16.838%	1283.285	0.594%
cap7*		0.001	0.001	0.002		0.023		0.047	
cap10*		0.001	0.001	0.003		0.024		0.073	
cap13*		0.002	0.002	0.006		0.027		0.159	
capa-c		0.152	0.159	0.476		1.072		19.457	
MO*		0.008	0.008	0.027		0.067		1.842	
MP*		0.049	0.050	0.166		0.235		11.229	
MQ*		0.142	0.143	0.523		0.669		25.027	
MR*		0.464	0.476	2.136		1.825		79.391	

Table 10: Running times of the algorithms

Problem	LB	JMS			MYZ		
		Cost	Error	Time	Cost	Error	Time
500,10	798399	808090	1.214%	0.364	813800	1.929%	0.375
500,100	326754	330758	1.225%	0.359	331183	1.355%	0.364
500,1000	99099	99245	0.147%	0.353	99208	0.110%	0.362
1000,10	1432737	1448286	1.085%	1.758	1454031	1.486%	1.810
1000,100	607591	613182	0.920%	1.758	613070	0.902%	1.799
1000,1000	220479	221942	0.664%	1.767	221437	0.435%	1.802
1500,10	1997302	2029316	1.603%	4.428	2032631	1.769%	4.543
1500,100	866231	875247	1.041%	4.409	875828	1.108%	4.467
1500,1000	334859	337307	0.731%	4.472	337015	0.644%	4.541
2000,10	2556794	2587422	1.198%	8.651	2587945	1.218%	8.921
2000,100	1122455	1133639	0.996%	8.516	1140205	1.581%	8.684
2000,1000	437553	441283	0.852%	8.599	441269	0.849%	8.730
2500,10	3095135	3142386	1.527%	14.514	3150382	1.785%	14.544
2500,100	1346924	1365831	1.404%	14.383	1369077	1.645%	14.548
2500,1000	534147	538463	0.808%	14.605	537891	0.701%	14.852
3000,10	3567125	3620604	1.499%	21.924	3637504	1.973%	22.310
3000,100	1600551	1618821	1.141%	21.660	1624535	1.498%	22.008
3000,1000	643265	648977	0.888%	21.630	649422	0.957%	21.914

Table 11: Results for the deterministic algorithms

For each set of points, we generated 3 instances. We set all opening costs to $\sqrt{n}/10$, $\sqrt{n}/100$ and $\sqrt{n}/1000$. Each opening cost defines a different instance with different properties.
In the following Tables we report the results of our experiments. In column 'LB' of Table 11 we provide the lower bound on each problem calculated by V&RR. In Tables 11 and 12 we report for each algorithm the average cost of the final solution, the average error and the average execution time of the algorithm. All errors were calculated using the lower bound in 'LB'.

Problem	LOCAL			TABU			V&RR		
	Cost	Error	Time	Cost	Error	Time	Cost	Error	Time
500,10	802178.35	0.473%	2.240	800478.70	0.260%	0.991	830634.10	4.037%	66.727
500,100	329126.95	0.726%	4.790	328539.60	0.546%	1.214	333459.55	2.052%	55.659
500,1000	99374.25	0.277%	2.167	99324.70	0.227%	1.755	104756.45	5.709%	43.099
1000,10	1439284.80	0.457%	17.282	1439905.55	0.500%	4.713	1532623.90	6.972%	356.955
1000,100	609825.80	0.358%	48.320	609577.65	0.327%	5.166	636226.65	4.713%	245.225
1000,1000	221736.45	0.570%	62.622	224990.50	2.046%	2.856	230848.05	4.703%	187.718
1500,10	2008847.75	0.578%	38.097	2005876.60	0.429%	10.288	2182858.00	9.290%	719.641
1500,100	870231.25	0.462%	150.230	870181.70	0.456%	11.285	903989.40	4.359%	556.186
1500,1000	336950.35	0.625%	257.636	336263.10	0.419%	18.786	347227.65	3.694%	479.491
2000,10	2570347.80	0.530%	85.863	2570231.45	0.526%	16.797	2804650.45	9.694%	1172.350
2000,100	1128591.55	0.547%	289.785	1128392.40	0.529%	18.803	1197988.15	6.729%	1000.169
2000,1000	439874.60	0.531%	682.241	439597.15	0.467%	88.423	452279.35	3.366%	912.588
2500,10	3114457.80	0.624%	196.352	3118274.75	0.748%	28.344	3414448.30	10.317%	2224.532
2500,100	1353003.85	0.451%	429.241	1352321.90	0.401%	29.282	1452854.95	7.865%	1728.433
2500,1000	536890.20	0.514%	1297.672	536545.95	0.449%	50.488	555853.25	4.064%	1401.563
3000,10	3586598.90	0.546%	228.680	3586916.35	0.555%	39.209	4018137.40	12.644%	2951.249
3000,100	1611474.10	0.682%	892.870	1611186.25	0.664%	44.901	1773741.80	10.821%	2677.263
3000,1000	646277.00	0.468%	2188.568	645680.15	0.375%	67.246	670984.45	4.309%	2008.729

Table 12: Results for the heuristic and randomized algorithms

On these metric benchmarks JMS again delivers slightly better results than MYZ. TABU is the best algorithm in terms of solution quality. LOCAL manages to find better solutions than the deterministic algorithms, but it is much slower than TABU, JMS and MYZ. The performance of V&RR is not competitive in comparison to the other algorithms. It is outperformed in terms of solution quality and execution time by all algorithms on nearly all benchmarks. Only on the larger benchmarks with small opening costs the running time of LOCAL is equally slow. Part of the reason for this is the use of priority queues. For the problems with smaller opening cost optimal solutions have a high number of opened facilities. Here the operations on the queues are getting expensive. A better implementation of LOCAL for these kinds of problems would omit the use of queues. Then the adjustment of the datastructures when opening a facility (which is the operation used more often here) could be executed in $O(m)$. The closing operation would need $O(nm)$, which leads to inferior execution times on average. However, here most of the time the closing operation is used in the exchange step, which is invoked after nearly all facilities have been opened. When nearly all facilities are opened, most of the cities are connected to the facility located at the same site. Then closing a facility affects basically only one city. In this case finding the new closest and second closest facilities can be done in $O(m)$. Thus, it is not surprising that an implementation without queues was able to improve the execution times on the large problems with $n = m > 1500$ by factors of up to 3. Nevertheless we chose to implement priority queues in our version of LOCAL as their theoretical advantage leads to shorter execution times on average.

3.5 Scaling and Local Search

In [11] a scaling technique was proposed to improve the approximation bound of local search for the metric UFLP. In the beginning all costs are scaled up by a factor of $\sqrt{2}$. Then the search is run on the scaled instance. Of all candidates found the algorithm exits with the one having the smallest cost for the unscaled instance.
With this technique the search is advised to open the most economical facilities. In practice, however, we cannot guarantee that the scaled local search picks better solutions. It is quite likely that the scaled version exits with inferior solutions as the solution space of the scaled instance might not reveal the same properties as the unscaled instance. Especially because the errors for the benchmarks are far lower than the approximation guarantee, it becomes obvious that this adjustment is only a way of lowering theoretical bounds and has limited practical use.
In Table 13 we report experimental results on a selection of benchmarks for the two versions of Local Search. The scaling technique method was proposed for Local Search on the metric UFLP. However, it deteriorates the performance of Local Search on metric as well as non-metric instances.

Bench	LOCAL			Scaled Local		
	Opt	Error	Time	Opt	Error	Time
Galvão and Raggi						
50	9,[99.5]	0.236%	0.006	2,[90.0]	0.484%	0.006
70	7,[80.0]	0.063%	0.013	0,[0.0]	0.704%	0.011
100	4,[86.3]	0.022%	0.026	0,[0.0]	0.650%	0.025
150	5,[92.0]	0.020%	0.062	0,[0.0]	0.682%	0.059
200	6,[65.8]	0.022%	0.127	1,[7.0]	0.819%	0.115
ORLIB						
cap71-74	2,[100]	0.044%	0.002	0,[0.0]	0.659%	0.002
cap101-104	2,[100]	0.059%	0.003	1,[100]	0.373%	0.004
cap131-134	2,[100]	0.236%	0.006	0,[0.0]	0.746%	0.006
capa-c	2,[75.0]	0.242%	0.476	1,[85.0]	1.243%	0.445
M*						
MO*	4,[78.8]	0.320%	0.027	2,[100]	0.720%	0.027
MP*	4,[85.0]	0.086%	0.166	2,[50.0]	0.822%	0.161
MQ*	5,[93.0]	0.041%	0.523	4,[36.3]	0.533%	0.590
MR*$^+$	5,[69.0]	0.229%	2.136	2,[32.5]	0.877%	2.513
MS$^+$	1,[100]	0.000%	11.720	0,[0.0]	0.666%	12.170
MT$^+$	1,[20.0]	0.159%	89.592	0,[0.0]	0.784%	118.443
k-median						
1000,10†		0.457%	17.282		1.416%	15.166
1000,100†		0.358%	48.320		1.663%	40.412
1000,1000†		0.570%	62.622		2.438%	75.220
2000,10†		0.530%	85.863		1.519%	99.594
2000,100†		0.547%	289.785		1.506%	246.756
2000,1000†		0.531%	682.241		2.074%	648.900
3000,10†		0.546%	228.680		1.459%	205.410
3000,100†		0.682%	892.870		1.489%	684.214
3000,1000†		0.468%	2188.568		2.011%	1816.916

$^+$ Error and Opt regarding best found solutions

† Error and Opt regarding lower bound by V&RR

Table 13: Results for the versions of Local Search

4 Conclusions

The uncapacitated facility location problem was solved by 5 different algorithms from different areas of optimization research. The deterministic algorithms manage to find good solutions on the benchmarks in short running times. Generally MYZ can improve the performance of JMS to the expense of little extra running time. On the tested metric instances the performance of the algorithms is competitive to the heuristic and randomized algorithms tested while the running times remain significantly shorter. Here JMS offers slightly better solution than MYZ. The approximation algorithms reveal higher errors only on a few tested non-metric instances, but always deliver solutions that are within 5% of optimum.
The presented Local Search profits mainly from the intelligent use of datastructures. On a number of instances the running times are able to compete with those of MYZ and JMS. However, due to the changing starting points the algorithm is not very robust. Scaling techniques that lead to improved approximation factors deteriorate the performance of the algorithm in practice. The tested version of the Volume algorithm V&RR is not competitive regarding solution quality and execution times.
TABU offers the best overall performance. In most cases TABU is able to find the optimal solution. It is much faster than V&RR (and Local Search on large-scale instances), but generally the running times cannot compete with those of MYZ and JMS.
All algorithms show a very good performance on the UFLP. TABU achieves best solution quality in a reasonable amount of time. It therefore should be the method of choice for practitioners.
Finally, we present a graphical chart with the results of the algorithms compared to the results of TABU. In Figure 1 we charted the results for the different benchmarks. The y-coordinates are calculated by the solution cost found by the algorithm divided by the solution cost found by TABU. x-coordinates are calculated accordingly with execution times. The times and costs were taken from the tables presented above. For the Bilde-Krarup Dq- and Eq-instances we averaged the results over all instances as well as for the `cap`- and M* instances. For the k-median problems we averaged the results over instances of the same size.
There are hardly any algorithms that have dots in the lower half of the plot. This indicates that there has been no algorithm to constantly outperform TABU in terms of solution cost. Moreover, there is hardly any dot in the lower left quadrangle. Dots in this region would indicate that TABU was outperformed in terms of solution cost and running times. The deterministic algorithms were faster than TABU, therefore there are some dots of JMS and MYZ in the upper left quadrangle with an x-coordinate less than 1. Bad performances are plotted in the upper right quadrangle. The dots in this region indicate that algorithms ended with solutions of bad quality and needed a high execution time. A lot of the dots of V&RR are located

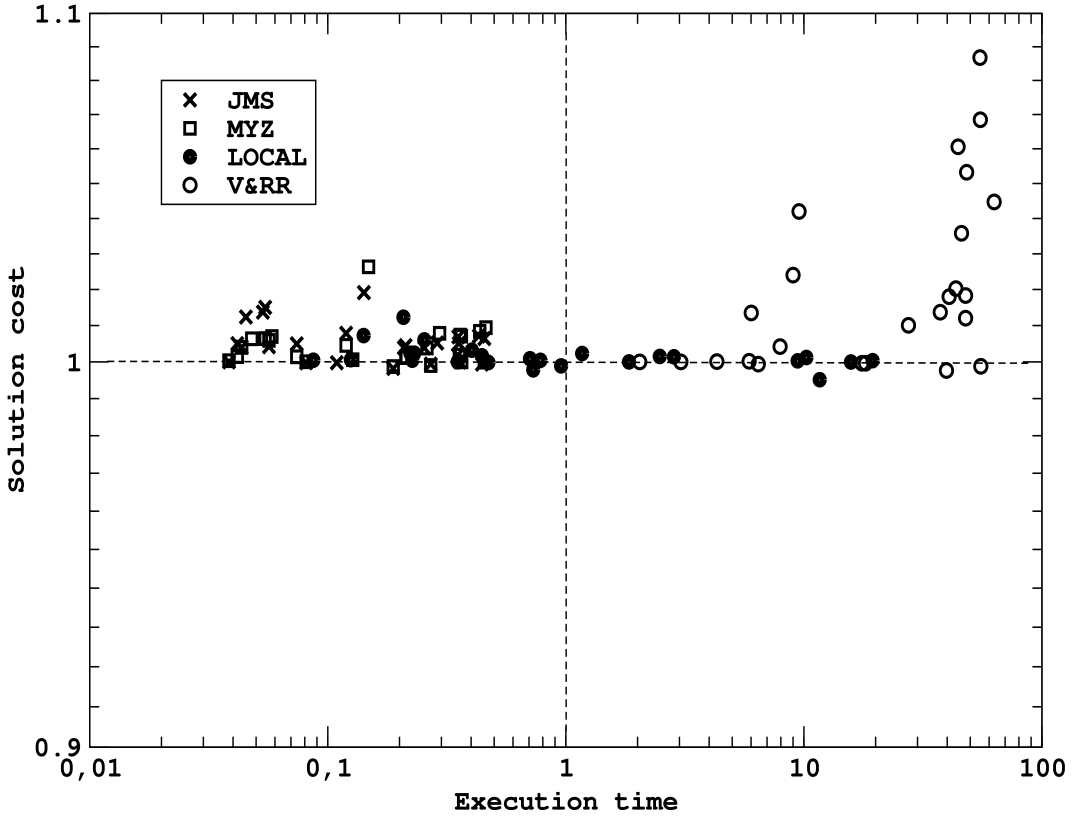

Figure 1: Plot of solution cost and execution times in comparison to TABU

here indicating the poor performance of this algorithm in comparison to TABU. LOCAL performed faster than TABU on small instances, but it was slower on the large instances. The solution cost was in most cases slightly worse in comparison to TABU. Therefore the dots are spread above the line in the upper half of the plot.

Acknowledgement

The author would like to thank Tobias Polzin for helpful hints and advice in the development of this study.

References

[1] S. Ahn, C. Cooper, G. Cornuéjols and A.M. Frieze. Probabilistic analysis of a relaxation for the k-median problem. *Mathematics of Operations Research*, 13:1-31, 1988.

[2] K.S. Al-Sultan and M.A. Al-Fawzan. A tabu search approach to the uncapacitated facility location problem. *Annals of Operations Research*, 86:91-103, 1999.

[3] M.L. Alves and M.T. Almeida. Simulated annealing algorithm for simple plant location problems. *Rev. Invest.*, 12, 1992.

[4] V. Arya, N. Garg, R. Khandekar, A. Meyerson, K. Munagala and V. Pandit. Local search heuristics for k-median and facility location problems. *ACM Symposium on Theory of Computing*, pages 21-29, 2001.

[5] M.L. Balinski. Integer programming: methods, uses, computation. *Management Science*, 12(3):253-313, 1965.

[6] F. Barahona. An implementation of the Volume algorithm. IBM COIN-OR website, *http://oss.software.ibm.com/developerworks/opensource/coin*, 2000.

[7] F. Barahona and R. Anbil. The volume algorithm: producing primal solutions with the subgradient method. Technical Report, IBM Watson Research Center, 1998.

[8] F. Barahona and F.A. Chudak. Near-optimal solutions to large scale facility location problems. Technical Report, IBM Watson Research Center, 2000.

[9] J.E. Beasley. Obtaining Test Problems via Internet. *Journal of Global Optimization*, 8:429-433, 1996.

[10] O. Bilde and J. Krarup. Sharp lower bounds and efficient algorithms for the simple plant location problem. *Annals of Discrete Mathematics*, 1:79-97, 1977.

[11] M. Charikar and S. Guha. Improved combinatorial algorithms for the facility location and k-median problems. In *Proceedings of the 40th Annual Symposium on Foundations of Computer Science*, 1999.

[12] F.A. Chudak. Improved approximation algorithms for uncapacitated facility location. In *Proceedings of the 6th IPCO Conference*, pages 180-194, 1998.

[13] R.D. Galvão and L.A. Raggi A method for solving to optimality uncapacitated facility location problems. *Annals of Operations Research*, 18:225-244, 1989.

[14] S. Guha and S. Khuller. Greedy strikes back: Improved facility location algorithms. *Journal of Algorithms*, 31:228-248, 1999.

[15] M.R. Korupolu, C.G. Plaxton and R. Rajaraman. Analysis of a local search heuristic for facility location problems. In *Proceedings of the 9th ACM-SIAM Symposium on Discrete Algorithms*, pages 1-10, 1998.

[16] J. Kratica, V. Filipovic, V. Sesum and D. Tosic. Solving the uncapacitated warehouse location problem using a simple genetic algorithm. In *Proceedings of the XIV International Conference on Material Handling and Warehousing*, pages 3.33-3.37, 1996.

[17] J. Kratica, D. Tosic and V. Filipovic. Solving the uncapacitated warehouse location problem by sga with add-heuristic. In *XV ECPD International Conference on Material Handling and Warehousing*, 1998.

[18] J. Kratica, D. Tosic, V. Filipovic and I. Ljubic. Solving the simple plant location problem by genetic algorithm. *RAIRO Operations Research*, 35:127-142. 2001.

[19] K. Jain, M. Mahdian and A. Sabieri. A new greedy approach for facility location problems. In *Proceedings of the 34th ACM Symposium on Theory of Computing*, forthcoming, 2002.

[20] K. Jain and V.V. Vazirani. Approximation algorithms for metric facility location and k-median problems using the primal-dual schema and langrangian relaxation. *Journal of the ACM*, 48:274-296, 2001.

[21] M. Mahdian, E. Marakakis, A. Sabieri and V.V. Vazirani. A greedy facility location algorithm analyzed using dual fitting. In *Proceedings of 5th International Workshop on Randomization and Approximation Techniques in Computer Science, Lecture Notes in Computer Science v. 2129*, pages 127-133. Springer-Verlag, 2001.

[22] M. Mahdian, Y. Ye and J. Zhang. Improved approximation algorithms for metric facility location problems. In *Proceedings of the 5th APPROX Conference*, forthcoming, 2002.

[23] P. Van Hentenryck and L. Michel. A simple tabu search for warehouse location. Technical Report, CS-02-05, Brown University, 2002.

[24] M. Sviridenko. An Improved Approximation Algorithm for the Metric Uncapacitated Facility Location Problem. In *Proceedings of the 10th IPCO Conference, Lecture Notes in Computer Science v. 2337*, pages 230 - 239, 2002.

[25] M. Sun. A Tabu Search Heuristic Procedure for the Uncapacitated Facility Location Problem. In C. Rego and B. Alidaee (eds.) *Adaptive Memory and Evolution: Tabu Search and Scatter Search*, Kluwer Academic Publishers, forthcoming, 2002.

[26] UflLib. UFLP-benchmarks, optimization code and benchmark generators. `http://www.mpi-sb.mpg.de/units/ag1/projects/benchmarks/UflLib`, 2002.

I N F O R M A T I K

Below you find a list of the most recent technical reports of the Max-Planck-Institut für Informatik. They are available by anonymous ftp from `ftp.mpi-sb.mpg.de` under the directory `pub/papers/reports`. Most of the reports are also accessible via WWW using the URL `http://www.mpi-sb.mpg.de`. If you have any questions concerning ftp or WWW access, please contact `reports@mpi-sb.mpg.de`. Paper copies (which are not necessarily free of charge) can be ordered either by regular mail or by e-mail at the address below.

Max-Planck-Institut für Informatik
Library
attn. Anja Becker
Stuhlsatzenhausweg 85
66123 Saarbrücken
GERMANY
e-mail: `library@mpi-sb.mpg.de`

MPI-I-2002-4-002	F. Drago, W. Martens, K. Myszkowski, H. Seidel	Perceptual Evaluation of Tone Mapping Operators with Regard to Similarity and Preference
MPI-I-2002-4-001	M. Goesele, J. Kautz, J. Lang, H.P.A. Lensch, H. Seidel	Tutorial Notes ACM SM 02 A Framework for the Acquisition, Processing and Interactive Display of High Quality 3D Models
MPI-I-2002-2-008	W. Charatonik, J. Talbot	Atomic Set Constraints with Projection
MPI-I-2002-2-007	W. Charatonik, H. Ganzinger	Symposium on the Effectiveness of Logic in Computer Science in Honour of Moshe Vardi
MPI-I-2002-1-008	P. Sanders, J.L. Träff	The Factor Algorithm for All-to-all Communication on Clusters of SMP Nodes
MPI-I-2002-1-005	T. Polzin	?
MPI-I-2002-1-004	S. Hert, T. Polzin, L. Kettner, G. Schäfer	Exp Lab A Tool Set for Computational Experiments
MPI-I-2002-1-003	I. Katriel, P. Sanders, J.L. Träff	A Practical Minimum Scanning Tree Algorithm Using the Cycle Property
MPI-I-2002-1-002	F. Grandoni	Incrementally maintaining the number of l-cliques
MPI-I-2002-1-001	T. Polzin, S. Vahdati	Using (sub)graphs of small width for solving the Steiner problem
MPI-I-2001-4-005	H.P.A. Lensch, M. Goesele, H. Seidel	A Framework for the Acquisition, Processing and Interactive Display of High Quality 3D Models
MPI-I-2001-4-004	S.W. Choi, H. Seidel	Linear One-sided Stability of MAT for Weakly Injective Domain
MPI-I-2001-4-003	K. Daubert, W. Heidrich, J. Kautz, J. Dischler, H. Seidel	Efficient Light Transport Using Precomputed Visibility
MPI-I-2001-4-002	H.P.A. Lensch, J. Kautz, M. Goesele, H. Seidel	A Framework for the Acquisition, Processing, Transmission, and Interactive Display of High Quality 3D Models on the Web
MPI-I-2001-4-001	H.P.A. Lensch, J. Kautz, M. Goesele, W. Heidrich, H. Seidel	Image-Based Reconstruction of Spatially Varying Materials
MPI-I-2001-2-006	H. Nivelle, S. Schulz	Proceeding of the Second International Workshop of the Implementation of Logics
MPI-I-2001-2-005	V. Sofronie-Stokkermans	Resolution-based decision procedures for the universal theory of some classes of distributive lattices with operators
MPI-I-2001-2-004	H. de Nivelle	Translation of Resolution Proofs into Higher Order Natural Deduction using Type Theory
MPI-I-2001-2-003	S. Vorobyov	Experiments with Iterative Improvement Algorithms on Completely Unimodel Hypercubes

MPI-I-2001-2-002	P. Maier	A Set-Theoretic Framework for Assume-Guarantee Reasoning
MPI-I-2001-2-001	U. Waldmann	Superposition and Chaining for Totally Ordered Divisible Abelian Groups
MPI-I-2001-1-007	T. Polzin, S. Vahdati	Extending Reduction Techniques for the Steiner Tree Problem: A Combination of Alternative-and Bound-Based Approaches
MPI-I-2001-1-006	T. Polzin, S. Vahdati	Partitioning Techniques for the Steiner Problem
MPI-I-2001-1-005	T. Polzin, S. Vahdati	On Steiner Trees and Minimum Spanning Trees in Hypergraphs
MPI-I-2001-1-004	S. Hert, M. Hoffmann, L. Kettner, S. Pion, M. Seel	An Adaptable and Extensible Geometry Kernel
MPI-I-2001-1-003	M. Seel	Implementation of Planar Nef Polyhedra
MPI-I-2001-1-002	U. Meyer	Directed Single-Source Shortest-Paths in Linear Average-Case Time
MPI-I-2001-1-001	P. Krysta	Approximating Minimum Size 1,2-Connected Networks
MPI-I-2000-4-003	S.W. Choi, H. Seidel	Hyperbolic Hausdorff Distance for Medial Axis Transform
MPI-I-2000-4-002	L.P. Kobbelt, S. Bischoff, K. Kähler, R. Schneider, M. Botsch, C. Rössl, J. Vorsatz	Geometric Modeling Based on Polygonal Meshes
MPI-I-2000-4-001	J. Kautz, W. Heidrich, K. Daubert	Bump Map Shadows for OpenGL Rendering
MPI-I-2000-2-001	F. Eisenbrand	Short Vectors of Planar Lattices Via Continued Fractions
MPI-I-2000-1-005	M. Seel, K. Mehlhorn	Infimaximal Frames: A Technique for Making Lines Look Like Segments
MPI-I-2000-1-004	K. Mehlhorn, S. Schirra	Generalized and improved constructive separation bound for real algebraic expressions
MPI-I-2000-1-003	P. Fatourou	Low-Contention Depth-First Scheduling of Parallel Computations with Synchronization Variables
MPI-I-2000-1-002	R. Beier, J. Sibeyn	A Powerful Heuristic for Telephone Gossiping
MPI-I-2000-1-001	E. Althaus, O. Kohlbacher, H. Lenhof, P. Müller	A branch and cut algorithm for the optimal solution of the side-chain placement problem
MPI-I-1999-4-001	J. Haber, H. Seidel	A Framework for Evaluating the Quality of Lossy Image Compression
MPI-I-1999-3-005	T.A. Henzinger, J. Raskin, P. Schobbens	Axioms for Real-Time Logics
MPI-I-1999-3-004	J. Raskin, P. Schobbens	Proving a conjecture of Andreka on temporal logic
MPI-I-1999-3-003	T.A. Henzinger, J. Raskin, P. Schobbens	Fully Decidable Logics, Automata and Classical Theories for Defining Regular Real-Time Languages
MPI-I-1999-3-002	J. Raskin, P. Schobbens	The Logic of Event Clocks
MPI-I-1999-3-001	S. Vorobyov	New Lower Bounds for the Expressiveness and the Higher-Order Matching Problem in the Simply Typed Lambda Calculus
MPI-I-1999-2-008	A. Bockmayr, F. Eisenbrand	Cutting Planes and the Elementary Closure in Fixed Dimension
MPI-I-1999-2-007	G. Delzanno, J. Raskin	Symbolic Representation of Upward-closed Sets
MPI-I-1999-2-006	A. Nonnengart	A Deductive Model Checking Approach for Hybrid Systems
MPI-I-1999-2-005	J. Wu	Symmetries in Logic Programs
MPI-I-1999-2-004	V. Cortier, H. Ganzinger, F. Jacquemard, M. Veanes	Decidable fragments of simultaneous rigid reachability
MPI-I-1999-2-003	U. Waldmann	Cancellative Superposition Decides the Theory of Divisible Torsion-Free Abelian Groups
MPI-I-1999-2-001	W. Charatonik	Automata on DAG Representations of Finite Trees
MPI-I-1999-1-007	C. Burnikel, K. Mehlhorn, M. Seel	A simple way to recognize a correct Voronoi diagram of line segments
MPI-I-1999-1-006	M. Nissen	Integration of Graph Iterators into LEDA

MPI-I-1999-1-005	J.F. Sibeyn	Ultimate Parallel List Ranking ?
MPI-I-1999-1-004	M. Nissen, K. Weihe	How generic language extensions enable "open-world" desing in Java
MPI-I-1999-1-003	P. Sanders, S. Egner, J. Korst	Fast Concurrent Access to Parallel Disks