Experimental comparison of heuristic and approximation algorithms for uncapacitated facility location

Martin Hoefer

Max-Planck-Institut für Informatik, Stuhlsatzenhausweg 85,
66123 Saarbrücken, Germany
mhoefer@mpi-sb.mpg.de
TU Clausthal, Marktstrasse 1,
38678 Clausthal-Zellerfeld, Germany
Martin.Hoefer@tu-clausthal.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 consider five of these approaches - 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 execution 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.

1 Introduction

The problem of locating facilities and connecting clients at minimum cost has been studied widely in Operations Research. In this paper we focus on the uncapacitated facility location problem (UFLP). 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.
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.

1.1 Approximation Algorithms

Recently some new 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 ([11], [24]), greedy augmentation ([10]) or primal-dual methods ([20], [10]). In terms of computational hardness Guha and Khuller [13] showed that it is impossible to achieve an approximation guarantee of 1.463 unless $NP \subseteq DTIME[n^{O(\log \log n)}]$. For our comparison we chose two of the newest and most promising algorithms.

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 Saberi [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 $O(n'^2\ log\ n')$ with $n' = \max\{n, m\}$. 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.

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 at present the best known factor 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

1.2 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.

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 for the experiments. It uses a slightly different representation of the problem. For a solution of the UFLP it is sufficient 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 (tabu-active) 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 $\text{cost}(S^*) = \infty$
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** $(\text{cost}(S) < \text{cost}(S^*))$ 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 an execution time of $O(m \log n)$ in each iteration.

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], [10]. In this paper we use the simple version of Arya et al [4], for which the authors could prove an approximation factor of 3 on metric instances. The algorithm works with the set S of opened facilities as a solution. An operation `op` is defined as opening or closing a facility or exchanging the status of an opened and a closed facility. To improve the execution 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 the first exchange move found that would give enough improvement. If there is no such move left, the algorithm stops. This 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

In our experiments the parameters were set to $\epsilon = 0.1$ and $p(n, m) = n + m$. Arya et al. suggested that the algorithm should be combined with the standard scaling techniques [10] 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. More on the unscaled and scaled versions of Local Search can be found in section 2.5 and [14].

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 [6]. 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 [7] used the rounding heuristic (RRWC) presented in [11] to find good upper bounds on the optimal dual solution cost and thereby reduced 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 involves 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 [5]. Regarding solution quality and execution 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 [5], [6], [7] for the specific details of the algorithm and the implementation.

2 Experiments

We tested all given algorithms on several sets of benchmark instances. First we studied the Bilde-Krarup benchmarks, which were proposed in [9]. These are non-metric small scale instances with $n \times m = 30 \times 80$ - 50×100. We chose this set because it is randomly generated, non-metric and involves the notion of increasing opening costs also present in the large scale k-median instances. Next we focused on small scale benchmarks proposed by Galvão and Raggi in [12]. These are metric instances with $n = m = 50$ - 200, which we chose because they make use of the shortest path metric and a Normal distribution to generate costs. Then we examined the performance on the `cap` instances from the ORLIB [8] and the M* instances, which were proposed in [18]. These are non-metric small, medium and large scale instances with $n \times m = 16 \times 50$ - 2000×2000. They have previously been used to examine the performance of many heuristic algorithms. Finally we studied metric large scale instances with $n = m = 1000$ - 3000, which were proposed in [1] and used as UFLP benchmarks for testing the performance of the Volume algorithm in [7]. So our collection of benchmark instances covers a variety of different properties: small, medium and large size; Euklidian metric, shortest path metric and non-metric costs; randomly generated costs from uniform distributions and Normal distributions.
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. Here we only report average results for the different benchmark types. For more detailed results of our experiments and values for the single instances the reader is referred to [14].
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].

2.1 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 3 we report the results of the runs for each algorithm.

Table 1. Parameters for the Bilde-Krarup problem classes

Type	Size ($n \times m$)	f_i	c_{ij}
B	50×100	Discrete Uniform (1000, 10000)	Discrete Uniform (0,1000)
C	50×100	Discrete Uniform (1000, 2000)	Discrete Uniform (0,1000)
Dq*	30×80	Identical, 1000*q	Discrete Uniform (0,1000)
Eq*	50×100	Identical, 1000*q	Discrete Uniform (0,1000)

* q=1,...,10

In columns 'Opt' we report the percentage of runs that ended with an optimal solution. In columns 'Error' we report the average error of the final solution in percentage of the optimal solution, in columns 'Time' the average execution time in seconds. In column 'CPX' we denoted the average execution time of CPLEX to solve the instances.
The deterministic algorithms perform quite good on these instances. The average error is 2.607% at maximum although the problems are not of metric nature. MYZ performs significantly better than JMS in terms of solution quality. It can solve additional 37 problems to optimality and 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 execution time it is much faster though. V&RR is outperformed by any of the other algorithms. It reveals the highest execution time and the worst solution quality.

2.2 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 δ = 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 2. The exact instances for these benchmarks are not known. As for the Bilde-Krarup benchmarks we generated 10 instances for each class. The results of our experiments are reported in Table 3. Columns 'Opt', 'Error' and 'Time' are defined as before. We also included the average execution times of CPLEX in column 'CPX'.
JMS performs slightly better than MYZ on these metric instances. Of the heuris-

Table 2. Parameters for the Galvão-Raggi problem classes

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

tic and randomized algorithms V&RR performs very good - even better than TABU and LOCAL - 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. LOCAL performs a little bit better than TABU, because both the execution times and the average errors are smaller. However, it is not very reliable to find optimal solutions.

2.3 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 4 we report the results for the different algorithms. In columns 'Opt' we again denote the percentage of runs that ended with an optimal solution. In Columns 'Error' we report the average error of the final solution in percentage of the optimal solution. 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. In columns 'Time' we report average execution times in seconds. Furthermore in column 'CPX' we report the average execution time of CPLEX.
Again the deterministic algorithms perform very well. 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 execution 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 again is 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. Hence, our results are consistent with the values reported in [23]. However, the execution times of our code are significantly faster than the times needed by the implementation of Michel and Van Hentenryck on a similar computer (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 execution times, however, are very competitive, as it performs in most cases significantly better than TABU.
V&RR performs generally worse than the other algorithms. On some of the cap instances it 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 small problems should therefore be avoided. For problems with $m, n \geq 100$, however, execution times of CPLEX become significantly higher.

2.4 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 [7] 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. 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 [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.
In Table 4 we report the results of our experiments for $n = 1000, 2000, 3000$. In column 'LB' we provide the lower bound on each problem calculated by V&RR. For each algorithm we report the average error and the average execution time. All errors were calculated using the lower bound in 'LB'.
On these metric benchmarks JMS again performs slightly better 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 execution times of LOCAL are equally slow. One reason for this is the use of priority queues. For the problems with smaller opening cost optimal solutions open a high number of facilities. Here the operations on the queues are getting expensive. When implemented without queues 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, in this case the closing operation is most often used in the

exchange step, which is called after nearly all facilities have been opened. Then most of the cities are connected to the facility located at the same site, and closing a facility affects only one city. So 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 $n = m > 1000$ 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.

2.5 Scaling and Local Search

In [10] 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 executed 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. However, the solution space of the scaled instance might not be similar to the space of the unscaled instance. Therefore in practice it is likely that the scaled version ends with inferior solutions. It becomes obvious that this adjustment is just for lowering theoretical bounds and has limited practical use. The scaling technique 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. Please see [14] for experimental results.

3 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 execution times. Generally MYZ can improve the performance of JMS to the expense of little extra execution time. On the tested metric instances the performance of the algorithms is competitive to the heuristic and randomized algorithms tested while the execution times remain significantly shorter. Here JMS offers slightly better solutions 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 from the intelligent use of datastructures. On a number of instances the execution times are able to compete with those of MYZ and JMS, but due to 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. TABU is able to find optimal solutions in most cases. It is much faster than V&RR (and Local Search on special instances), but the execution times cannot compete with those of MYZ and JMS. The tested version of the Volume algorithm V&RR is not competitive regarding solution quality and execution times.
The preference for a method in practice depends on the properties of the problem

instances and the setting. If speed is most important, JMS or MYZ should be used, especially if metric instances are to be solved. If solution quality is most important, TABU should be used. In a general setting the results indicate a preference for TABU, as it achieves best solution quality in a reasonable amount of time.
Finally we present a plot of the results in relation to TABU. The x- and y-coordinates represent values regarding execution time and solution cost, respectively. The coordinates were calculated by dividing the results of the algorithms by the results of TABU. We further adjusted some of the data by averaging over the D- and E-instances of Bilde-Krarup and the instances of the same size of k-median, respectively. There are 22 dots for each algorithm.
Only a few dots are located in the lower half of the plot, i.e. hardly any time TABU was outperformed in terms of solution cost. Moreover, there is hardly any dot in the lower left quadrangle, which indicates better performance in terms of execution time and solution cost. In the upper left quadrangle most of the dots of JMS and MYZ are located indicating faster performance with higher solution costs. In the upper right part most of the dots of V&RR are located. This means worse performance regarding execution time and solution cost. Most of the dots of LOCAL are spread closely above the line in the upper half, which is due to slightly higher solution costs, the faster performance on smaller and the slower performance on larger instances.

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

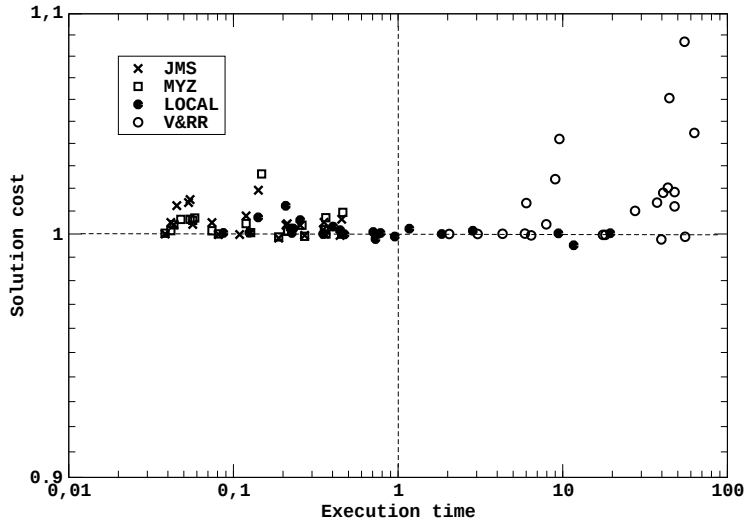

Fig. 1. Plot of solution costs and execution times in comparison to TABU

Table 3. Results for Bilde-Krarup and Galvão-Raggi instances

Type	CPX	JMS			MYZ			LOCAL			TABU			V&RR		
		Opt	Error	Time	Opt	Error	Time	Opt	Error	Time	Opt	Error	Time	Opt	Error	Time
B	6.859	50%	0.416	0.003	40%	0.588	0.003	80%	0.046	0.012	100%	0.000	0.053	70%	0.419	0.421
C	107.558	10%	1.750	0.003	30%	0.886	0.003	42%	0.848	0.014	64%	0.245	0.055	1%	4.454	0.525
D1	21.591	0%	2.445	0.001	10%	1.689	0.002	25%	1.678	0.006	54%	0.241	0.038	2%	3.719	0.239
D2	30.990	10%	1.675	0.002	20%	1.133	0.002	20%	1.758	0.006	67%	0.537	0.044	5%	3.083	0.254
D3	28.103	10%	2.607	0.002	40%	0.923	0.002	29%	0.879	0.006	89%	0.073	0.042	9%	2.245	0.235
D4	26.685	30%	0.796	0.002	30%	0.597	0.002	72%	0.530	0.006	100%	0.000	0.041	32%	1.248	0.243
D5	22.368	40%	0.647	0.002	70%	0.085	0.002	59%	0.402	0.006	94%	0.004	0.040	38%	0.995	0.246
D6	28.393	20%	1.042	0.002	30%	1.315	0.002	50%	0.882	0.006	87%	0.146	0.042	42%	0.919	0.259
D7	24.484	10%	1.771	0.002	60%	0.664	0.002	80%	0.354	0.005	100%	0.000	0.042	80%	0.214	0.251
D8	20.947	40%	1.587	0.002	40%	1.044	0.002	63%	1.000	0.006	90%	0.166	0.043	40%	1.390	0.259
D9	22.326	70%	0.846	0.002	90%	0.012	0.002	80%	0.285	0.006	100%	0.000	0.043	73%	0.496	0.256
D10	19.122	70%	0.252	0.002	80%	0.189	0.002	63%	0.760	0.006	93%	0.139	0.043	74%	0.506	0.268
E1	133.839	20%	2.265	0.003	30%	1.317	0.003	10%	1.430	0.013	57%	0.388	0.062	0%	5.712	0.516
E2	229.305	20%	1.650	0.003	40%	0.845	0.003	14%	2.712	0.013	94%	0.006	0.067	0%	4.479	0.560
E3	190.860	20%	1.610	0.003	20%	0.940	0.003	38%	0.784	0.012	63%	0.268	0.061	5%	3.419	0.553
E4	185.168	30%	1.192	0.003	30%	0.781	0.004	24%	1.577	0.013	90%	0.013	0.060	5%	2.505	0.581
E5	163.571	10%	2.560	0.003	70%	0.690	0.004	30%	2.019	0.013	100%	0.000	0.062	21%	1.924	0.546
E6	173.918	40%	1.049	0.003	50%	0.661	0.004	56%	0.969	0.013	100%	0.000	0.062	22%	1.981	0.602
E7	164.845	50%	0.759	0.004	50%	0.613	0.004	52%	0.996	0.015	100%	0.000	0.063	12%	1.802	0.586
E8	180.186	10%	1.474	0.004	40%	0.887	0.004	40%	1.043	0.014	90%	0.177	0.067	37%	1.318	0.585
E9	174.150	30%	1.232	0.004	60%	0.674	0.004	65%	0.655	0.013	100%	0.000	0.066	42%	0.896	0.592
E10	148.229	40%	0.775	0.004	60%	0.404	0.004	74%	0.948	0.013	100%	0.000	0.066	50%	0.864	0.598
50	0.200	100%	0.000	0.001	100%	0.032	0.001	90%	0.236	0.006	100%	0.000	0.026	97%	0.007	0.112
70	0.332	90%	0.038	0.003	70%	0.065	0.003	56%	0.063	0.013	90%	0.061	0.037	93%	0.001	0.238
100	0.677	90%	0.014	0.006	80%	0.099	0.007	35%	0.022	0.026	84%	0.039	0.055	90%	0.002	0.965
150	1.623	70%	0.059	0.016	60%	0.111	0.016	46%	0.020	0.062	50%	0.239	0.085	85%	0.001	3.375
200	3.355	60%	0.071	0.036	70%	0.032	0.036	40%	0.022	0.127	49%	0.131	0.133	68%	0.011	7.363

Table 4. Results for ORLIB, M and k-median instances

Type	CPX	JMS			MYZ			LOCAL			TABU			V&RR		
		Opt	Error	Time	Opt	Error	Time	Opt	Error	Time	Opt	Error	Time	Opt	Error	Time
cap7*	0.107	0%	0.401	0.001	10%	0.397	0.001	50%	0.044	0.002	100%	0.000	0.023	100%	0.000	0.047
cap10*	0.098	0%	0.499	0.001	10%	0.144	0.001	50%	0.059	0.003	100%	0.000	0.024	100%	0.000	0.073
cap13*	0.196	0%	0.493	0.002	10%	0.138	0.002	50%	0.236	0.006	100%	0.000	0.027	85%	0.017	0.159
capa-c	77.745	0%	1.990	0.152	0%	2.712	0.159	50%	0.242	0.476	82%	0.077	1.072	70%	0.039	19.457
MO*		20%	0.786	0.008	30%	0.452	0.008	63%	0.320	0.027	100%	0.000	0.067	23%	1.004	1.842
MP*		20%	0.387	0.049	30%	0.118	0.050	68%	0.086	0.166	100%	0.000	0.235	6%	1.203	11.229
MQ*		20%	0.444	0.142	30%	0.132	0.143	93%	0.041	0.523	100%	0.000	0.669	4%	1.367	25.027
MR*		[10%]	[0.381]	0.464	[10%]	[0.380]	0.476	[69%]	[0.229]	2.136	[100%]	[0.000]	1.825	0%	[2.022]	79.391
MS		[10%]	[0.000]	2.281	[10%]	[0.000]	2.323	[100%]	[0.000]	11.720	[100%]	[0.000]	6.366	0%	[1.829]	304.066
MT		0%	[0.205]	11.079	0%	[0.205]	11.241	[100%]	[0.159]	89.529	[90%]	[0.020]	31.505	0%	[1.817]	1283.285

Problem	LB	JMS		MYZ		LOCAL		TABU		V&RR	
		Error	Time	Error	Time	Error	Time	Error	Time	Error	Time
1000,10	1432737	1.085	1.758	1.486	1.810	0.457	17.282	0.500	4.713	6.972	356.955
1000,100	607591	0.920	1.758	0.902	1.799	0.358	48.320	0.327	5.166	4.713	245.225
1000,1000	220479	0.664	1.767	0.435	1.802	0.570	62.622	2.046	2.856	4.703	187.718
2000,10	2556794	1.198	8.651	1.218	8.921	0.530	85.863	0.526	16.797	9.694	1172.350
2000,100	1122455	0.996	8.516	1.581	8.684	0.547	289.785	0.529	18.803	6.729	1000.169
2000,1000	437553	0.852	8.599	0.849	8.730	0.531	682.241	0.467	88.423	3.366	912.588
3000,10	3567125	1.499	21.924	1.973	22.310	0.546	228.680	0.555	39.209	12.644	2951.249
3000,100	1600551	1.141	21.660	1.498	22.008	0.682	892.870	0.664	44.901	10.821	2677.263
3000,1000	643265	0.888	21.630	0.957	21.914	0.468	2188.568	0.375	67.246	4.309	2008.729

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. F. Barahona. An implementation of the Volume algorithm. IBM COIN-OR website, *http://oss.software.ibm.com/developerworks/opensource/coin*, 2000.
6. F. Barahona and R. Anbil. The Volume algorithm: producing primal solutions with the subgradient method. Technical Report, IBM Watson Research Center, 1998.
7. F. Barahona and F.A. Chudak. Near-optimal solutions to large scale facility location problems. Technical Report, IBM Watson Research Center, 2000.
8. J.E. Beasley. Obtaining Test Problems via Internet. *Journal of Global Optimization*, 8:429-433, 1996.
9. 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.
10. 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.
11. F.A. Chudak. Improved approximation algorithms for uncapacitated facility location. In *Proceedings of the 6th IPCO Conference*, pages 180-194, 1998.
12. 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.
13. S. Guha and S. Khuller. Greedy strikes back: Improved facility location algorithms. *Journal of Algorithms*, 31:228-248, 1999.
14. M. Hoefer Performance of heuristic and approximation algorithms for the uncapacitated facility location problem. Research Report MPI-I-2002-1-005, Max-Planck-Institut für Informatik, 2002.
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. Saberi. A new greedy approach for facility location problems. In *Proceedings of the 34th Symposium on Theory of Computing 2002*, 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, Lecture Notes in Computer Science v. 2462*, pages 229 - 242, 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.