An Acceleration of Exact Algorithms for the Uncapacitated Facility Location Problem

Jaroslav Janáček[1], Ľuboš Buzna[1,2]

[1]Department of Transportation Networks, Faculty of Management Science and Informatics, University of Žilina, Univerzitná 8215/1, SK-010 26 Žilina , Slovak Republic
e-mail: jardo@frdsa.uniza.sk, buzna@frdsa.uniza.sk

[2]Institute of Transport & Economics, Faculty of Traffic and Transport Sciences, Dresden University of Technology, Andreas-Schubert-Str. 23, D-01062 Dresden, Germany
e-mail buzna@vwi.tu-dresden.de

Abstract

This contribution is focused on an acceleration of branch and bound algorithms for the uncapacitated facility location problem. Our approach is based on the well - known Erlenkotters' procedures and Körkels' multi - ascent and multi - adjustment algorithms, which have proved to be the efficient tools for solving the large - sized instances of the uncapacitated facility location problem. These two original approaches were examined and a thorough analysis of their performance revealed how each particular procedure contributes to the computational time of the whole algorithms. These analyses helped us to focus our effort on the most frequent procedures. The unique contribution of this paper is a new dual ascent procedure. This procedure leads to considerable acceleration of the lower bound computation process and reduces the resulting computational time. To demonstrate more efficient performance of amended algorithms we present the results of extensive numerical experiments.

Keywords: branch and bound, uncapacitated facility location problem, dual ascent algorithm, DualLoc*,* PDLoc

1 Introduction

Location science is a well-established field in applied mathematics and in operation research. Classification and broad overview of location problems and solution techniques can be found, for example, in Hale and Moberg (2003), Brandeau and Chiu (1989), Owen and Daskin (1998), Klose and Drexl (2005) and Labbé and Louveaux (1997). Location science comprises the vast amount of problem specifications and their variations. One of the basic but important problems is the uncapacitated facility location problem (UFLP) introduced in Balinski (1965). The uncapacitated facility location problem belongs to the most popular topics of theoretical research and computational analysis in the last four decades (Galvão (2004)). Its attractiveness originates from several characteristics of the problem. Although the problem is known to be NP-hard (Cornuejols et al. (1990)), there was soon found an effective exact algorithm, which enables to solve large-sized instances of the problem. This property is very rare in the family of integer programming problems.

The other fact, which contributes to the attractiveness of this problem, is its broad applicability (Janáček (2004)). The solving technique can be used not only for a design of the cost optimal structure of two-echelon distribution system, but it can be also embedded into various algorithms, which were designed to solve more complex location problems. As reported in (Galvão (2004)), it is possible to reformulate the classical maximum distance problem, p – median problem, and the maximum covering location problem into a form of the UFLP. Furthermore, the algorithm for the UFLP can be employed in an approximate solving of the p – centre problem. These circumstances motivated us to focus our research on improving the current exact solving techniques for UFLP.

The UFLP consists of placing facilities such as warehouses, offices or hospitals in some sites of a given finite set I and in a consecutive unique assignment of customers from a given finite set J to the located facilities. The term "customer" can denote a real customer, but it can also stand for an inhabitant of some region, a patient in a hospital, or a retailer and so on. The possible operations, i.e. the facility location and the customer assignment, should minimize the value of a cost objective function. This objective function includes both the fixed charges f_i paid for facility location at the location i and the costs c_{ij} for demand satisfaction of the *j-th* customer from the location i.

Defining the variables $y_i \in \{0,1\}$ for $i \in I$ (where $y_i = 1$ means placing a new facility at location i) and the variables $z_{ij} \in \{0,1\}$ for $i \in I$ and $j \in J$ (where $z_{ij} = 1$ represents the assignment of location i to customer j) we get the following model of the UFLP:

$$\text{Minimize} \quad z_P = \sum_{i\in I} f_i y_i + \sum_{i\in I} \sum_{j\in J} c_{ij} z_{ij} \qquad (1)$$

$$\text{Subject to} \quad \sum_{i\in I} z_{ij} = 1 \quad \text{for } j\in J \qquad (2)$$

$$z_{ij} \le y_i \quad \text{for } i \in I \text{ and } j \in J \qquad (3)$$

$$y_i,\ z_{ij} \in \{0, 1\} \quad \text{for } i \in I \text{ and } j \in J. \qquad (4)$$

Many authors have dealt with this problem (Körkel (1989), Conn and Cornuéjols (1990) and Goldengorin et al. (2004)). Nevertheless, the now seminal procedure, DualLoc, proposed by Erlenkotter (1978) remains one of the most effective algorithms, and enables solutions in tractable times (Crainic (2002)). Inspired by this approach, the exact algorithms PDLoc (Körkel (1989)) and BBDual (Janáček and Kovačiková (1997)) were implemented and tested. These algorithms exploit the relation between the primary and dual formulation of the strong linear programming relaxation of the original problem. And when needed, the procedures, calculating the dual and induced primary solution, are followed by branch and bound method. The duality gap was analytically studied by Mladenović et al. (2006). Conn and Cornuéjols (1990) proposed an orthogonal projection method to solve the dual problem to optimality. The Boolean representation of the UFLP allows to construct the rules for reduction of the UFLP instances, and either to solve or reduce the size of solved subproblems (Mladenović et al. (2006), Körkel (1989), Goldengorin et al. (2004)). Such rules were tested in Goldengorin et al. (2004), showing the significant reduction of the computational time. The data correcting method (Goldengorin et al. (2003)) allows to determine the exact and approximate solutions of the UFLP. This method is applicable to broad spectrum of problems with super modular objective function and performs very well also for UFLP, when combined with elements of the Erlenkotter's approach.

The combinatorial background of the UFLP enables simple, but a very efficient application of modern heuristics. Alves and Almeida (1992) successfully used the simulated annealing method. A genetic algorithm was employed in Kratica et al., (2001) and tabu search strategies were used to solve UFLP in Michel and Hentenrych (2001) and in Sultan and Fawzan (1999). In Ghosh (2003) is compared the performance of several steepest descent local search heuristics using various neighbourhood structures. Choosing the best performing structure the tabu search strategy and the complete local search strategy were implemented. In summary, Ghosh concluded a very good performance of studied algorithms on UFLP of smaller and medium sizes (up to 750 customers and 750 facility locations). The hybrid and multi-start heuristic, combining the characteristics of several metaheuristics, was applied in Resende and Werneck (2006) giving very good results for large set of numerical experiments.

The remainder of our paper is organized as follows. In Section 2, we briefly introduce the existing exact solving methods PDLoc and BBDual. Section 3 presents the results of preliminary experiments, which identify the critical points of both algorithms. The suggested improvements are then described in Section 4.

The benefits of our rearrangements are reported in Section 5. To conclude this paper, in Section 6, we summarize our findings and suggest some possible directions of further research.

2 Dual based lower bounding in branch and bound methods for UFLP

The basic idea, which was originally introduced in the algorithm DualLoc and which was also followed by the algorithms BBDual and PDLoc, consists in relation between linear relaxation of the original problem (1) – (4) and the associated dual problem.

After some reformulation and introduction of slack variables s_i, the dual problem (see Körkel (1989)) takes the following form:

$$\text{Maximize} \quad z_D = \sum_{j \in J} v_j \qquad (5)$$

$$\text{Subject to} \quad \sum_{j \in J} \max\{0, v_j - c_{ij}\} + s_i = f_i \qquad \text{for } i \in I \quad (6)$$

$$s_i \geq 0 \qquad \text{for } i \in I \quad (7)$$

Objective function value z_D of any feasible solution is smaller or equal to any objective function value z_P of any feasible solution of the linear relaxation of (1) - (4), according to the weak duality theorem. Thus, objective function value of arbitrary feasible solution of (6) - (7) constitutes a lower bound for optimal solution of the problem (1) - (4).

To obtain a good lower bound from the dual problem, Erlenkotter (1978) suggested combination of two procedures. The first of them, the dual ascent algorithm (DA), introduced by Bilde and Krarup (1977), starts from an arbitrary feasible solution of the dual problem and subsequently increases the values of the v_j variables as long as constraints (6) and (7) hold.

The second procedure enables a further improvement of the dual solution obtained by the DA procedure. The dual adjustment procedure (DAD) searches for a configuration, in which a decrease of some variable v_j by value δ will create free space at least at two locations i and i' from I, which can be used for an ascent of at least two different variables v_k and v_l ($k \neq l \neq j$). Each found configuration is exploited and followed by the dual ascent procedure. Having obtained a dual solution, an induced primal feasible solution can be obtained by applying the complementary constraints (8) – (10):

$$(\max\{0, v_j - c_{ij}\})z_{ij} = 0 \qquad \text{for } i \in I, j \in J \qquad (8)$$

$$s_i y_i = 0 \qquad \text{for } i \in I \qquad (9)$$

$$(y_i - z_{ij}) \max\{0, v_j - c_{ij}\} = 0 \qquad \text{for } i \in I, j \in J \qquad (10)$$

The original PRIMA procedure ensures the validity of constraints (8) – (9*)* and constructs the associated primal solution to the dual solution such that the constraints (10) are violated as slightly as possible. All these procedures are described in more detail in Erlenkotter (1978), Körkel (1989) and Janáček and Kovačiková (1997).

2.1 Algorithm BBDual

If the variables y_i are fixed, the optimal values of variables z_{ij} can be easily found. It is sufficient to assign the customer j to the facility i, for which the value of coefficient c_{ij} is minimal. Thus, the most complex problem is to determine the setting of the variables y_i. The BBDual algorithm (see Janáček and Kovačiková (1997)) is based on the branch and bound method, in which branching is performed by fixing the variables y_i to zeros or ones.

The algorithm makes use of the depth first strategy. To decide if a given branch should be processed or excluded from the searching process, a lower bound of high quality is needed. Such lower bound can be obtained by successively performing the dual ascent and dual adjustment algorithms. These procedures provide dual feasible solution and the corresponding value of objective function serves as the searched lower bound. Furthermore, a corresponding primal feasible solution is generated. This is done by the PRIMA procedure, which follows the complementary conditions (8) – (10) for both the primal and the dual problem. The best-found primal solution is stored and its objective function value constitutes an upper bound of the optimal solution.

2.2 Algorithm PDLoc

The algorithms PDLoc and BBDual are built up on the same principles, which were introduced in the original algorithm DualLoc. However, the PDLoc algorithm comprehends a number of effective modifications and improvements on the original procedures and, in addition, it is enhanced by several new procedures. Similar to the BBDual algorithm, the PDLoc employs the branch and bound method to determine the optimal solution, but in contrast to the BBDual, the strategy of the lowest lower bound is used in the searching tree processing.

Varying the order of customers in the PRIMA procedure, enables to open new locations each time and to explore a broader spectrum of primal solutions. This leads to a faster decrease of the upper bound and to the faster termination of searching process.

In the case where the fixed charges f_i are considerably higher than the allocation costs c_{ij}, the first dual solution is calculated by the dual multi-ascent procedure (MDA) instead of the dual ascent procedure. In the dual multi-ascent procedure, the variables v_j are initially set to some high value and then they are adjusted to meet constraints (7). Executing the original dual ascent procedure completes this process. This rearrangement gives a considerable time saving as shown in the next sections.

Other improvement consists of applying the simple exchange heuristic after the first primal solution is found. Moreover, when a big difference between the upper and lower bound occurs, it is reduced by the modified dual adjustment procedure. This procedure consists of two phases. The first of them is called the primal-dual multi adjustment (PDMAdj) and the second is the primal-dual adjustment (PDAdj). In the first phase, the values of variables v_j are extensively reduced and subsequently gradually incremented using a modified dual ascent algorithm. In the second phase, the values of variables v_j are reduced in the loop, variable by variable, and then the resulting dual solution is processed by the dual ascent algorithm. The number of repetitions of the dual adjustment procedure depends on the problem size.

The used branch and bound searching scheme allows to fix the selected locations to be permanently opened ($y_i = 1$) or closed ($y_i = 0$) and thereby to reduce the size of the solved problem. To fix a variable, special conditions have to be satisfied. Evaluation of these conditions is time consuming, especially being deep in the searching three. Therefore the fixing of variables is preferred in the root of the searching tree (pre-processing). If a processed branch is deep in the searching tree, the variables are fixed only if there is a possibility to fix several variables simultaneously. For more detailed explanations we refer the reader to the original work Körkel (1989).

3 Experimental evaluation of algorithms BBDual and PDLoc

We implemented both algorithms using the integrated development environment Delphi. In our implementation we restricted the values of the coefficients f_i and c_{ij} to integers.

Following our main goal to improve the computational properties of both algorithms, we started with a broad set of numerical experiments, testing the algorithms BBDual and PDLoc on a set of problems generated from the real-world transportation networks. The experiments were not focused only on revealing the computational properties of the whole algorithms, but we studied also the computational complexity of particular procedures. It enabled us to compare their importance and mutual interactions. Special attention was paid to the dependence of computational time on the size of the problem and on the ratio between the coefficients f_i and c_{ij}.

3.1 Benchmarks

Three sets of testing problems were used to perform the above-mentioned numerical experiments. The first set is formed by the well-known Beasley's testing problems (Beasley (1990)) cap41, …, cap134, the sizes of which ($|I| \times |J|$) vary from 16×50 to 50×50 including three testing problems capa, capb and capc of size 100×1000. The values of coefficients in objective function were rounded to integers.

Since the standard Beasley's testing problems are relatively small we have generated two other sets, named K90 and G700. The set K90 consists of 90 testing problems derived from the railway network of Slovak Republic. These medium sized problems are ordered in ten groups: 45×457, 91×457, 137×457, 182×457, 229×457, 274×457, 319×457, 365×457, 411×457 and 457×457. The set G700 consists of 700 problems derived from the road network of Slovak Republic. This set consists from ten subgroups, sizes of which ranged from 100×2906, 200×2906, as large as 1000×2906. Each subgroup contained 70 benchmarks. For each size of the benchmark 10 different random subgraphs of the road network graph of corresponding size were generated. Each subgraph was used as a base for creating seven benchmarks by modifying the coefficients c_{ij} and f_i to cover uniformly the whole spectrum of located facilities in optimal solution. For instance, for a problem of size 100×2906 the optimal cardinality of located facilities were 1, 17, 33, 50, 66, 83 and 100 respectively. The source code of the algorithms and the used benchmarks are available from the authors upon request.

3.2 Preliminary results of numerical experiments

We solved all problems by both algorithms to obtain the frequency in which the particular procedures are called. Together with computational time and its distribution over the procedures, we also evaluated the numbers of visited branches in the branch and bound method. All numerical experiments mentioned in this paper were performed on a PC equipped with Intel 2.4 GHz processor, 256 MB RAM and the computational time was measured with preciseness of at least 60 ms.

The BBDual algorithm solved the Beasley's problems cap41, …, cap134 in less than 60 ms. The problems capa, capb and capc needed 0.5, 5.44 and 3.13 seconds, respectively. The PDLoc algorithm solved the problems cap41, …, cap134 in an average time of 85 ms and the larger problems capa, capb and capc in 1.91, 47.18 and 107.16 seconds, respectively.

The average computation times for problem sets G700 and K90 are listed in Table 1 and Table 2, respectively.

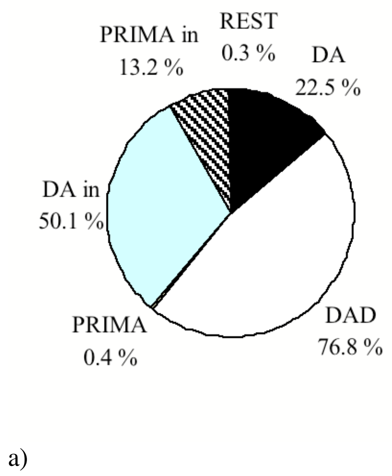
Table 1 Average time in seconds and corresponding standard deviation for the class of benchmarks G700

Size of problems	*BBDual*		*PDLoc*	
	t [*s*]	*stdD*	*t* [*s*]	*stdD*
100x2906	5.343	12.64	1.44	0.93
200x2906	27.16	68.33	1.80	0.99
300x2906	52.95	143.11	2.40	1.48
400x2906	127.06	337.58	2.74	1.82
500x2906	134.17	340.52	5.29	6.52
600x2906	277.59	700.73	5.21	5.54
700x2906	277.70	704.57	6.12	5.45
800x2906	497.26	1248.42	8.56	8.87
900x2906	640.44	1652.65	11.45	11.25
1000x2906	644.88	1595.72	10.60	11.19

Table 2 Average time in seconds and corresponding standard deviation for the class of benchmarks K90

Size of problems	*BBDual*		*PDLoc*	
	t [*s*]	*stdD*	*t* [*s*]	*stdD*
45x457	0	0	0.13	0.05
91x457	0.05	1.64	0.29	0.24
137x457	0.14	34.6	0.57	0.64
182x457	0.18	31.6	0.56	0.54
229x457	1.11	5037.69	1.70	0.29
274x457	0.87	554.98	0.86	0.54
319x457	1.32	1467.69	1.23	1.05
365x457	1.57	577.08	1.33	0.55
411x457	2.59	5281.33	1.66	1.25
457x457	6.71	8961295.25	6.96	7.05

In addition to the total computational time we also measured the relative time consumed by each of the particular procedures. The averaged results, obtained from the experiments with the problem set G700, are shown in Figure 1. The abbreviations “DA”, “DAD” and “PRIMA” denote the relative average time consumed by the associated procedures. The “REST” includes the time spent by branching operations, lower bound computation, as well as necessary memory operations. The labels “Prima in“ and “DA in“ denote the time spent by procedures PRIMA and DA, which are called by other procedures (i.e., the time “DA in” includes the time spent by DA procedure when it has been called from the DAD procedure). This also explains why the sum over all procedures is larger than 100 %.



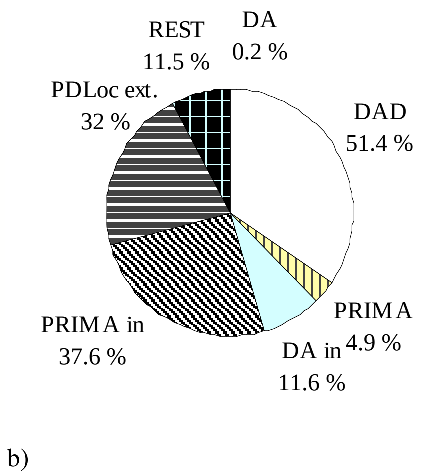


Fig. 1 The average distribution of computational time among the procedures of BBDual algorithm in (a) and PDLoc algorithm in (b) (these results were obtained for set G700)

Comparing the results listed in Table 1 and Table 2, it can be found that both tested algorithms reached almost the same average computational time solving the problems from the set K90. On the contrary, solving the large problems from the set G700 the BBDual algorithm was considerably slower. The standard deviation for the BBDual algorithm is also much higher. This can be explained by the inefficient performance of BBDual on a specific subset of solved problems. These problems can be characterized by the considerably higher values of f_i in comparison to the values c_{ij}. Such disproportion of coefficients causes the ineffective performance of algorithm DA in its first run. This delay did not occur in the PDLoc algorithm, since it was eliminated by the MDA procedure. The distribution of computation time over the basic procedures DA, DAD and PRIMA differs considerably. The BBDual algorithm spent in average 72.6% of the computational time by performing the DA procedure, while the PDLoc algorithm spent on this procedure only 11.8% of the total computational time.

The total time consumed by the DAD procedure is approximately the same for both algorithms, but its distribution among the time-consuming activities differs considerably. In the case of BBDual, the DAD procedure spent 50.1% of the time performing the DA procedure. In contrary, the PDLoc algorithm needed only a small part of the time (11.6%) for the DA procedure embedded in DAD algorithm. This implies that the PDLoc algorithm focuses on a more intensive searching for improving operations. As a consequence the average number of inspected branches in the branch and bound method decreases (see Table 3).

Table 3 The average number of inspected branches (*PV*), the minimal number of inspected branches (*Min PV*), the maximal number of inspected branches (*Max PV*) and the corresponding standard deviation (*Std PV*) (these results were obtained for set G700)

Size of problems	*BBDual*				*PDLoc*			
	PV	*Min PV*	*Max PV*	*Std PV*	*PV*	*Min PV*	*Max PV*	*Std PV*
100 x 2906	6.9	1	59	13.9	1	1	1	0.0
200 x 2906	18.5	1	169	40.6	1	1	1	0.0
300 x 2906	15.1	1	181	37.9	1.1	1	3	0.5
400 x 2906	27.0	1	389	69.7	1.1	1	3	0.5
500 x 2906	22.9	1	337	56.1	1.6	1	7	1.3
600 x 2906	37.3	1	429	94.5	1.5	1	7	1.3
700 x 2906	34.8	1	461	79.2	1.5	1	5	1.0
800 x 2906	58.9	1	470	119.7	2.2	1	9	2.1
900 x 2906	58.9	1	630	133.4	2.3	1	11	2.2
1000 x 2906	48.9	1	693	113.6	1.7	1	11	1.7

The performed analysis indicated the benefits of the MDA procedure, and also gave us a notion about the time consumed by particular procedures and allowed us to identify the most critical parts of the both algorithms. Moreover, the analysis helped us to estimate, how the intensity of searching for new improvements in the DAD procedure influences the number of visited branches in the branch and bound method.

4 Rearrangements of the algorithms BBDual and PDLoc

The preliminary experiments and the subsequent analysis of results showed that the DA procedure is the greatest time consumer as concerns the BBDual algorithm. The next finding, which follows from the analysis, is that the MDA procedure considerably improves the computational process when the algorithm PDLoc is used. Taking into account these two observations, we have focused our efforts on an improvement of the DA procedure and on an overall rearrangement of the BBDual algorithm utilizing the MDA procedure.

4.1 New scheme of the DA procedure

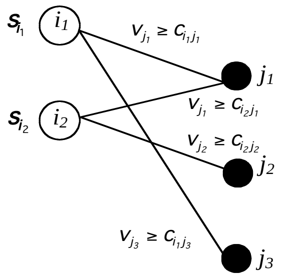
The original DA procedure (see Figure 2) processes the set of relevant customers J^+, customer by customer, in an order, which follows an input sequence of the relevant customers. At each step, the variable v_j corresponding to the customer $j \in J^+$, is incremented by a value d. The value d is determined as a maximal value, which satisfies the constraints (6). Hence, it is obvious, that this variable cannot be increased in the further steps, and thus the customer j can be excluded from the set J^+. This procedure is repeated until J^+ is empty.

```
**Procedure DA** ($J^+$)
While $|J^+| > 0$, do :
        Get next $j \in J^+$.
        Set $d_1 = \min_i s_i$ with $i \in \{i: c_{ij} \leq v_j\}$.
        Set $d_2 = \min_i (c_{ij} - v_j)$ with $i \in \{i: c_{ij} > v_j\}$.
        If $d_1 > d_2$, set $d = d_2$. Else, delete $j$ from $J^+$ and set $d = d_1$.
        If $d > 0$, then:
                For each $i \in \{i: c_{ij} \leq v_j\}$, set $s_i = s_i - d$.
                Set $v_j = v_j + d$ and $z_D = z_D + d$.
Terminate.
```

Fig. 2 The original DA procedure as it was introduced by Körkel (1989)

In the following, we will show that the performance of this procedure significantly depends on the order in which the set of relevant customers is processed. This drawback is explained on the example with two possible locations i_1, i_2 and three customers j_1, j_2 and j_3, depicted in Figure 3. Let us associate two slack variables s_{i_1} and s_{i_2} with the locations i_1 and i_2. An edge connecting a customer j to a location i denotes that the inequality $v_j \geq c_{ij}$ holds for this pair of subscripts. Denoting by symbol K_j the cardinality of set $\{i \in I: c_{ij} \geq v_j\}$ for a customer j, we get $K_{j_1} = 2$, $K_{j_2} = 1$ and $K_{j_3} = 1$.

The DA procedure processes the customers in the order, which is given by the sequence in which they entered the procedure (i.e. first to be processed is the customer j_1 followed by customers j_2 and j_3). If the processing of customer j_1 enables an increase of the variable v_{j_1} by value β, then the lower bound increases exactly by β. The maximal theoretical increase of the lower bound is given by the sum of s_{i_1} and s_{i_2}. In the example depicted in Figure 3, the increment β has to be subtracted from the both slack variables s_{i_1} and s_{i_2}, and it has to meet constraints (6). It means that the theoretical capacity $s_{i_1} + s_{i_2}$ is reduced by 2β in order to increase the lower bound by β.



s_{i_1}
i_1
s_{i_2}
i_2
$v_{j_1} \geq c_{i_1 j_1}$
$v_{j_1} \geq c_{i_2 j_1}$
$v_{j_2} \geq c_{i_2 j_2}$
$v_{j_3} \geq c_{i_1 j_3}$
j_1
j_2
j_3

Fig. 3 The setting of parameters showing the situation, when the ordering of customers improves the efficiency of DA procedure

Taking into consideration the theoretical capacity and its possible decrease by modifying the variables v_j we formulate a new DA* procedure, which exploits the potential for the increasing of the lower bound in a more efficient way. Our approach is based on prioritising the customers, which selection promises a larger increase of the lower bound z_D (5) in the next steps. We will order ascendingly the relevant customers J^+ according to cardinalities of K_j. The benefit of this scheme can be demonstrated on the example depicted in Figure 3.

Having applied the designed prioritisation rule, the customers are processed in the order j_2, j_3 and j_1. If processing of the customer j_2 enables an increase of the variable v_{j_2} by β, then considering constraints (6) only the slack variable s_2 has to be reduced by β. In this way, the theoretical capacity is reduced by value β and the lower bound, accordingly, increases by β. As demonstrated, this approach may reduce the sum of slack variables s_i less than the original DA procedure does, keeping the same increase of variables v_{j_2}.

```
Procedure DA* ($J^+$)
Order the set $J^+$ of relevant customers into a sequence $j_1, j_2, \dots j_k, \dots j_n$
ascendingly with respect to the cardinalities $K_j$.
For $k = 1$ to $|J^+|$ do:
    Set $j := j_k$.
    Set $d = min\{min\{s_i : i \in I, c_{ij} \leq v_j\}, \{c_{ij} - v_j : i \in I, c_{ij} > v_j\}\}$.
    If $d > 0$, then:
        Update $s_i$ for $i \in I$, $c_{ij} \leq v_j$ by $s_i := s_i - d$. Set $v_j := v_j + d$ and $z_D = z_D + d$.
        If the cardinality $K_j$ has increased, then reorder the sequence of
        customers $j_k, \dots, j_n$.
Terminate.
```

Fig. 4 New DA* procedure

4.2 Further rearrangement of the algorithms

Based on the above-mentioned analysis and the preliminary experiments we implemented and verified the following rearrangements:

- We applied the DA* procedure in the both algorithms.
- We embedded the MDA procedure into the BBDual algorithm to improve its efficiency in the cases, when the ratio of fixed charges f_i and costs c_{ij} is large.
- We amended the evaluation of branches in BBDual algorithm. Both newborn branches are evaluated simultaneously and the more perspective branch is processed first.

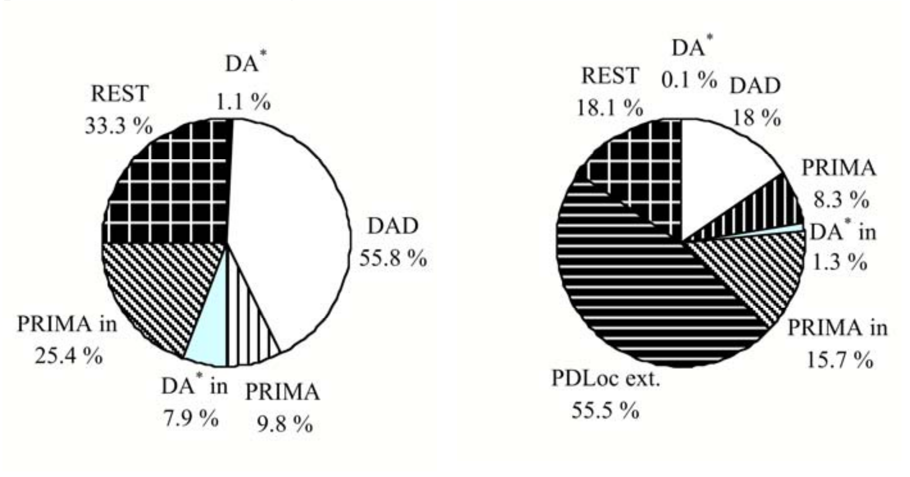
In this way, we formed new versions of the original algorithms BBDual and PDLoc. To distinguish the original and the new versions of algorithms, the modified algorithms are denoted as BBDual* and PDLoc*. It should be noted that all the above-described modifications were studied also separately. None of them brought larger improvements in the computational time as when used individually.

5 Results of numerical experiments

The effects of suggested rearrangements were extensively examined by numerical experiments performed on three sets of testing problems, described in

Section 3.1. Similarly to Section 3.2, we evaluated the computational time and its distribution among the particular procedures together with the number of inspected branches.

Figure 5 gives an evidence of a significant time reduction, when the DA* procedure is used. The total usage was reduced from 72.6 % to 9% in the BBDual* algorithm as well as from 11.8% to 1.4 % in the PDLoc* algorithm. This result suggests that our new DA* procedure has a significant influence on the performance of the both algorithms.



a) b)

Fig. 5 The average distribution of computational time among the procedures of BBDual* algorithm in (a) and PDLoc* algorithm in (b) (these results were obtained for set G700)

The BBDual* algorithm solved Beasley's testing problems cap41, …, cap134 in less than 60 ms. The problems capa, capb and capc were solved in 0.33, 1.32 and 1.87 seconds, respectively. The PDLoc* algorithm solved problems cap41, …, cap134 in an average time of 13 ms and the larger problems capa, capb and capc in 0.55, 0.55, and 19.77 seconds, respectively.

Comparison of these results with the performance of the original algorithms reported in Section 3.2, indicates that the proposed modifications brought considerable time savings.

In Table 4, we report the average computational time and the standard deviation in which the BBDual* and PDLoc* algorithms solved the testing problems G700. To facilitate the comparison with the original algorithms, Table 4 and Table 5 list the computational time of the original algorithms in the columns labelled BBDual and PDLoc.

Table 4 Average time in seconds and corresponding standard deviation for benchmarks G700

Size of problems	*BBDual*	*BBDual**		*PDLoc*	*PDLoc**	
	t [*s*]	*t* [*s*]	*StdD*	*t* [*s*]	*t* [*s*]	*StdD*
100x2906	5.343	0.28	0.29	1.44	0.77	0.34
200x2906	27.16	0.41	0.39	1.80	0.87	0.22
300x2906	52.95	0.74	0.64	2.40	1.20	0.90
400x2906	127.06	1.01	0.47	2.74	1.46	0.88
500x2906	134.17	1.75	1.05	5.29	2.83	0.90
600x2906	277.59	2.54	1.78	5.21	3.64	2.88
700x2906	277.70	3.90	2.77	6.12	4.63	4.38
800x2906	497.26	5.07	4.23	8.56	6.45	6.35
900x2906	640.44	7.24	6.31	11.45	8.89	8.83
1000x2906	644.88	7.07	5.89	10.60	7.47	8.08

Presented results show that our modifications led to considerable decrease of the overall computational time. We should note that the using of MDA procedure in algorithm BBDual also contributed to this significant reduction. A similar reduction of computational time was also observed, when the benchmarks K90 were processed (see Table 5).

Table 5 Average time in seconds and corresponding standard deviation for benchmarks K90

Size of problems	*BBDual*	*BBDual**		*PDLoc*	*PDLoc**	
	t [*s*]	*t* [*s*]	*StdD*	*t* [*s*]	*t* [*s*]	*StdD*
45x457	0	0.01	0.02	0.13	0.09	0.02
91x457	0.05	0.02	0.03	0.29	0.13	0.04
137x457	0.14	0.08	0.13	0.57	0.29	0.31
182x457	0.18	0.12	0.12	0.56	0.30	0.22
229x457	1.11	0.62	0.90	1.70	0.80	1.27
274x457	0.87	0.44	0.37	0.86	0.43	0.27
319x457	1.32	1.13	1.41	1.23	0.97	0.76
365x457	1.57	8.53	20.75	1.33	0.97	0.47
411x457	2.59	14.54	31.02	1.66	1.14	0.98
457x457	6.71	7.60	10.80	6.96	4.78	6.92

Table 6 reports on the average number of inspected branches, the minimal and maximal numbers of inspected branches and the standard deviation for the numbers of inspected branches. These outputs together with the computational time give evidence of the remarkable acceleration of the both algorithms.

Table 6 The average number of inspected branches (*PV*), the minimal number of the inspected branches (*Min PV*), the maximal number of inspected branches (*Max PV*) and the corresponding standard deviations (*Std PV*) (these results were obtained for set G700)

Size of problems	*BBDual*[*]				*PDLoc*[*]			
	PV	*Min PV*	*Max PV*	*Std PV*	*PV*	*Min PV*	*Max PV*	*Std PV*
100 x 2906	1.4	1	5	0.9	1	1	1	0
200 x 2906	1.2	1	3	0.6	1	1	1	0
300 x 2906	1.8	1	7	1.5	1.2	1	5	0.7
400 x 2906	1.5	1	7	1.2	1.0	1	3	0.2
500 x 2906	2.7	1	11	2.4	1.5	1	7	1.2
600 x 2906	3.1	1	17	3.4	1.3	1	5	0.8
700 x 2906	4.9	1	35	6.0	1.5	1	11	1.4
800 x 2906	6.2	1	47	8.5	2.1	1	9	2.1
900 x 2906	7.7	1	45	10.1	2.3	1	13	2.5
1000 x 2906	5.6	1	53	7.8	1.6	1	7	1.4

6 Conclusions

In this paper we described a detailed performance analysis of two exact algorithms BBDual and PDLoc for solving the UFLP. Our analysis revealed bottlenecks of these algorithms and helped us to identify the procedures, which contributed the most to the overall computational time. Moreover, this analysis suggested how the trade off between the intensity of searching process for new lower bound and the number of visited branches influences the computational time.

We proposed a new DA procedure, which exploits the duality slacks in a more efficient way and we implemented two other minor modifications to the BBDual algorithm. To study the effects of these modifications we created a large set of benchmarks based on real transportation networks. Our experiments confirmed the significant improvement of both algorithms. Applying these modifications, the BBDual[*] algorithm not only outperforms the original version of the PDLoc algorithm, but it performs even better than improved version PDLoc[*].

Our study of computation performance of particular procedures indicated the direction in which we could improve the properties of both algorithms even more. This idea is based on the classification of solved problems at the beginning of the solving process and in adaptation of the procedures and their parameters to the particular problem class. This issue will be a subject of our forthcoming investigations.

Acknowledgements

The authors are grateful for the financial support by the Ministry of Education of the Slovak Republic (project VEGA 1/3775/06) and thank two anonymous referees for comments and suggestions that greatly improved the manuscript.

References

Alves, M.L., & Almeida, M.T. (1992). Simulated annealing algorithm for simple plant location problem. *Rev. Invest.,* 12.

Balinski, M. (1965). Integer programming: Methods, uses computation. *Management Science*, 12, 254-313.

Beasley, J.E. (1990). OR-Library: distributing test problems by electronic mail. *Journal of the Operational Research Society,* 41, 1069-1072.

Bilde, O., & Krarup, J. (1997). Sharp lower bounds and efficient algorithms for the simple plant location problem. *Annals of Discrete Mathematics*, 1, pp 77-97.

Brandeau M.L., & Chiu, S.S. (1989). An overview of representative problems in location research. *Management Science*, 35, 6, 645-674.

Conn, A.R., & Cornuéjols, G. (1990). A Projection method for the uncapacitated facility location problem. *Mathematical Programming*, 46, pp. 273-298.

Cornuejols, G., Nemhauser, G.L. & Wolsey, L.A. (1990) The uncapacitated facility location problem. In: *Mirchandani, P.B., & Francis, R.L. (Eds.),* Discrete Location Theory, Wiley-InterScience, New York, pp. 119-171.

Crainic, T.G. (2003). Long-Haul Freight Transportation. Handbook of Transportation Science, Springer New York, NY.

Erlenkotter, D. (1978). A Dual-Based Procedure for Uncapacitated Facility Location. *Operations Research,* 26(6), 992-1009.

Galvão, R.D. (2004). Uncapacitated Facility Location Problems: Contributions. *Pesquisa Operacional,* 24, 7 -38.

Ghosh, D. (2003). Neighborhood search heuristics for the uncapacitated facility location problem. *European Journal of Operational Research*, 150, 150-162.

Goldengorin, B., Tijssen, G.A., Ghosh, D., & Sierksma G. (2003). Solving the Simple Plant Location Problem using a Data Correcting Approach. *Journal of Global Optimization,* 25(4), 377-406.

Goldengorin, B. Ghosh, D. & Sierksma, G. (2004). Branch and peg algorithms for the simple plant location problem. *Computers & OR* 31(2) pp. 241-255.

Hale, T.S., & Moberg, C.R. (2003). Location Science Research: A Review. *Annals of Operations Research,* 123, 21-35.

Janáček, J., & Kovačiková, J. (1997). Exact Solution Techniques for Large Location Problems. In *Proceedings of the International Conference: Mathematical Methods in Economics, Ostrava,* 80-84.

Janáček, J. (2004). Service System Design in the Public and Private Sectors. In *Proceedings of the International Conference: Mathematical Methods in Economics, Virt,* 101 – 108.

Klose, A., & Drexl, A. (2005). Facility location models for distribution system design. *European Journal of Operational Research* 162, 4-29.

Körkel, M. (1989). On then exact solution of large – scale simple plant location problem. *European Journal of Operational Research,* 39(2), 157-173.

Kratica, J., Tosic, D., Filipovic, V., & Ljubic, I. (2001). Solving the simple plant location problem by genetic algorithm *RAIRO Operations Research,* 35, 127-142.

Labbé, M., &Louveaux, F.V., (1997). Location problem. In: DellAmico N, Maffioli F and Martello S (Eds). Annotated Bibliographies in Combinatorial Optimization. John Wiley & Sons: New York, pp. 264–271.

Michel, L., & Hentenrych, P.V. (2004). A simple tabu search for warehouse location. *European Journal of Operational Research,* 157(3), 576 - 591.

Mladenović, N., Brimberg, J., & Hansen, P. (2006). A note on duality gap in the simple plant location problem. *European Journal of Operational Research*, 174, 11-22.

Owen, H.O., & Daskin, M.S. (1998). Strategic facility location: A review. *European journal of Operational Research* 111, 423-447.

Resende, M.G.C, & Werneck, R.F. (2006). A hybrid multistart heuristic for the uncapacitated facility location problem. *European Journal of Operational Research*, 174, 54-68.

Sultan, K.S., & Fawzan, M.A. (1999). A tabu search approach to the uncapacitated facility location problem. *Annals of Operations Research,* 86, 91 – 103.