An Efficient Exact Algorithm for the Vertex p-Center Problem and Computational Experiments for Different Set Covering Subproblems

Taylan Ilhan*
F. Aykut Özsoy†
Mustafa Ç. Pınar ‡

November 26, 2002

Abstract

We develop a simple and yet very efficient exact algorithm for the problem of locating p facilities and assigning clients to them in order to minimize the maximum distance between a client and the facility to which it is assigned. The algorithm iteratively sets a maximum distance value within which it tries to assign all clients, and thus solves set covering subproblems. Three different set covering problems are incorporated and excellent computational results are reported for each on an extensive list of test problems derived from OR-Lib and TSP-Lib problems with up to 900 data points. An initial phase of the algorithm consists in solving linear programming subproblems, and is shown experimentally to yield very tight lower bounds to the optimal value.

Key words. Integer programming, p-center problem, facility location.

1 Introduction

The purpose of this paper is to describe a simple, efficient and flexible algorithm which utilizes a set-covering subproblem for the solution of the vertex p-center problem. The algorithm is a two-stage variant of a well-known algorithm by Minieka [7] long considered inefficient. Three different set covering type subproblems are incorporated into the algorithm and extensive computational results are reported for each.

The p-center problem consists of locating p facilities and assigning clients to them so as to minimize the maximum distance between a client and the facility it is assigned to. The problem is known to be NP-hard [5]. A typical application is locating fire stations or ambulances, where the distance from the facilities to the farthest client is to be minimum. For a detailed exposition of the p-center problem and available solution methodology, the reader is directed to Chapter 5 of the textbook [2].

*E-mail: *t-ilhan@northwestern.edu* Department of Industrial Engineering & Management Sciences, Northwestern University, 60208 Evanston, Illinois, US.

†E-mail: *aykut@bilkent.edu.tr* Department of Industrial Engineering, Bilkent University, 06533 Ankara, Turkey.

‡Corresponding author. E-mail: *mustafap@bilkent.edu.tr*. Department of Industrial Engineering, Bilkent University, 06533 Ankara, Turkey.

Let $W = \{w_1, w_2, \ldots, w_m\}$ be the set of all possible locations for facilities with $|W| = M$, $V = \{v_1, v_2, \ldots, v_n\}$ be the set of all clients with $|V| = N$. The distance for each facility pair (w_i, v_j) is given as d_{ij}. We assume in this paper that $W \cup V$ is the vertex (node) set of a complete graph, and distances d_{ij} represent the length of the shortest path between vertices i and j (in the test problems of Section 3, we deal with a single vertex set, i.e., with the special case where $W = V$).

An integer programming formulation for the problem is the following (see, e.g., [2]):

$$
\begin{aligned}
&\text{Minimize} && z && && \\
&\text{subject to} && && && \\
&\sum_j x_{ij} && = && 1 \quad \forall i \in V && (1.1)\\
&x_{ij} && \leq && y_j \quad \forall i \in V, j \in W && (1.2)\\
&\sum_{j \in W} y_j && \leq && p \quad \forall i \in V && (1.3)\\
&\sum_{j \in W} d_{ij} x_{ij} && \leq && z && (1.4)\\
&x_{ij}, y_j && \in && \{0,1\} \quad \forall i \in V, \forall j \in W && (1.5)
\end{aligned}
$$

where x_{ij} assumes value one if client i is assigned to facility site j, and value zero otherwise. The binary variable y_j assumes value one if facility site j is open. Constraints (1.1) express the requirements that all clients must be assigned to some facility site. Constraints (1.2) prevent a client from an assignment to a facility site which is not open. In total at most p facility sites are to be opened, a requirement which is modeled by constraint (1.3).

The above integer programming formulation cannot be used for solving large instances. In 1970, Minieka proposed an exact solution procedure for the problem but his algorithm has also been considered to be inefficient since it solves successive instances of a minimal set covering problem, which is NP-complete itself on general graphs (see Chapter 4 of [2]). In this algorithm, Minieka initially chooses an arbitrary set of p facility sites from W, M_1, and finds the distance to the farthest client from this set of facilities (say ε_1). This value is used as the coverage distance of a minimal set covering problem where one checks whether at most p facilities can cover all clients within this coverage distance (Minieka's subproblem is explained in Section 2.2 in detail). If it turns out that all clients can not be covered within this distance, this means that M_1 is the optimal p-center set. Otherwise, the set of facility sites corresponding to $y_j = 1$ is named M_2; the distance to the farthest client from the facility sites in M_2 is computed, say ε_2 (where $\varepsilon_2 < \varepsilon_1$). Then the above-mentioned subproblem for ε_2 is solved, and the algorithm repeats these steps.

Effective heuristic procedures were also proposed for the problem recently in [8] where Mladenovic, Labbé and Hansen claim that "no efficient algorithm able to solve large instances (of the p-center problem) seems to exist up to now". However, two efficient and exact algorithms for the p-center problem emerged in the literature, due to Daskin [3], and to Elloumi, Labbe and Pochet[4], respectively. Both algorithms solve successive set covering subproblems, and rely on carrying out an iterative search over coverage distances rather than sets of facility sites. They differ in three ways from each other; subproblems they use, methods they utilize for solving subproblems efficiently, and the method of search for optimal objective value over coverage distances. Daskin has formulated a maximal set covering subproblem and solved it by Lagrangian relaxation (Daskin's subproblem is described in detail in Section 2.2). Elloumi et al. use Minieka's subproblem and find a solution to it by a greedy heuristic and by solving the IP formulation of the subproblem.

Against this background, the contribution of the present paper is to unify in a simple, efficient

and flexible algorithm the aforementioned ideas in a way that can capitalize on different subproblems proposed in the literature. In Section 2 we explain the idea underlying the proposed algorithm and describe in detail the subproblems used. In Section 3 we give our detailed computational results.

2 The Proposed Algorithm and Subproblems

2.1 Algorithm

Similar to algorithms by Daskin and Elloumi et al., our algorithm also relies on solving a series of set covering problems while carrying out an iterative search over the coverage distances. But, it is not dependent on a specific subproblem and can utilize different subproblems. At each iteration, we set a threshold distance as a radius, see whether it is possible to cover all clients with p or less facilities within this radius (i.e. we use this radius as the coverage distance of the set covering problem we use), and update lower and upper bounds on the optimal radius in the light of this information. Namely, the idea underlying our solution procedure is to proceed as follows: select initial lower and upper bounds on the optimal objective function value; solve an appropriate set covering problem by using the average of the lower and upper bounds as the coverage distance; if p or less facilities can cover all clients within this radius, reset the upper bound to the coverage distance that was just used; if not, reset the lower bound to the coverage distance just used. If the lower and upper bounds are equal, stop; otherwise set the coverage distance as the average of lower and upper bounds and continue the process.

To avoid solving successive IP's of set covering subproblems, our algorithm is designed as a two-phase variant of the above procedure, to speed up the process. In the first phase, a binary search for a suitable lower bound on the optimal value is carried out by solving LP relaxations of the subproblems. In the second phase subproblems are solved as IPs for increasing coverage distances starting from the bound provided by stage 1. The first coverage distance encountered in this process within which all clients can be covered with at most p facilities is the optimal objective value of the p-center problem.

The algorithm is as follows[1]:

Step 1. Find the minimum, l, and maximum, u, of weights of all edges,
$l = \min\{d_{ij} : \forall i \in V, \forall j \in W\}$
$u = \max\{d_{ij} : \forall i \in V, \forall j \in W\}$

Step 2. Calculate $dif = \lfloor (u-l)/2 \rfloor$, $\varepsilon = l + dif$.

Step 3. Solve the LP relaxation of the selected subproblem with coverage distance ε.

Step 3.1. If all clients can not be covered with at most p facilities within this coverage distance, then set $l = \varepsilon$.

Step 3.2. else set $u = \varepsilon$.

Step 4. Calculate $(u - l)$.

Step 4.1. If $(u - l) \leq 1$ then go to Step 5.

Step 4.2. else go to Step 2.

Step 5. If all clients could not be covered with at most p facilities within this coverage distance, then set $\varepsilon = u$, else set $\varepsilon = l$.

Step 6. Create the subproblem (in IP form) by setting coverage distance to ε and solve.

Step 6.1. If at most p facilities can not cover all clients, then increase the value of ε:
$\varepsilon' = \min\{d_{ij} : d_{ij} > \varepsilon, \forall i \in V, \forall j \in W\}$ then set $\varepsilon = \varepsilon'$ and go to Step 6.

Step 6.2. else Stop.

[1] This algorithm is for integer data set.

In the algorithm, Step 3.1, Step 5 and Step 6.1 are determined specifically with respect to the subproblem used, because, the condition 'if all clients can not be covered with at most p facilities within this coverage distance' has a different interpretation in each subproblem as explained in the next section.

The algorithm can be modified to be used for problems with non-integral distance values. For this purpose, Step 5 of the algorithm must be modified as
Step 5. Set $\varepsilon = min\{d_{ij} : d_{ij} \geq l\}$.

2.2 Subproblems

Two subproblems have been used in the literature for solving the p-center problem. One is the minimal set covering problem first used by Minieka [7] and the other is the maximal set covering problem of Daskin [3]. Together with these two subproblems, we will introduce a new one, the feasibility subproblem, and explain the required specifications in the algorithm (in steps 3.1, 5 and 6.1) for each.

We start with Minieka's set covering problem. This set covering problem minimizes the number of facilities to be located while covering all clients within a coverage distance. Formulation is as follows:

$$\begin{aligned} \text{Minimize} \quad & \sum_{j \in W} y_j \\ \text{subject to} \quad & \\ \sum_{j \in W : d_{ij} < \varepsilon_1} y_j \quad & \geq \quad 1 \qquad \forall i \in V \\ y_j \quad & \in \quad \{0,1\} \qquad \forall j \in W \end{aligned}$$

where y_j takes value 1 if a facility is located at site j, and 0 otherwise.

In his algorithm, Minieka seeks to cover each client by a facility whose distance to the client is 'strictly less than' the coverage distance. This leads to the result that at each iteration the set of p facility sites found yields a strictly smaller p-center objective function value (i.e., distance to the farthest client from the facility sites chosen) than the previous iteration. In other words, the set of p facility sites obtained at each iteration is a better solution to the p-center problem than the one from the previous iteration.

Different from Minieka's, our algorithm carries out iterative search over coverage distances as mentioned. In this context, we have made a small modification to Minieka's subproblem for using it in our algorithm. We allowed clients to be covered with facilities whose distance to the client is 'less than or equal to' the coverage distance. Namely, we changed the condition under summation from $\{j \in W : d_{ij} < \varepsilon\}$ to $\{j \in W : d_{ij} \leq \varepsilon\}$ in the formulation. Thus, the formulation of Minieka's subproblem as used in our algorithm has become

$$\begin{aligned} \text{Minimize} \quad & \sum_{j \in W} y_j \\ \text{subject to} \quad & \\ \sum_{j \in W : d_{ij} \leq \varepsilon} y_j \quad & \geq \quad 1 \qquad \forall i \in V \\ y_j \quad & \in \quad \{0,1\} \qquad \forall j \in W \end{aligned}$$

The following example explains the need for this change. Let $K_1, K_2, \ldots, K_n$ represent distinct values in the distance matrix (i.e., the matrix of d_{ij}'s) such that $K_1 < K_2 < \ldots < K_n$ (where $n \leq |V|^2$). Suppose the first phase of our algorithm dictates to start second phase with coverage distance equal to K_t $(t < n)$. This means that we will solve successive set covering problems with coverage distances $K_t, K_{t+1}, K_{t+2} \ldots$ respectively until we encounter one within which all clients can be covered with at most p facilities. Suppose optimal radius is K_{t+h}. If we use Minieka's subproblem in its original form, with coverage distance K_{t+h}, the optimal objective value of the subproblem is going to be more than p, and we will proceed to a new iteration and use $K_{t+(h+1)}$ as the coverage distance. In this iteration, the subproblem will give optimal objective less than or equal to p and the algorithm will stop. This means that we would solve the problem in $h+2$ second-phase iterations. However, if the change mentioned above is carried out, the subproblem with coverage distance K_{t+h} would yield the optimal objective value p (or less), and the algorithm would stop in $h+1$ second-phase iterations. Hence, the aforementioned modification saves one iteration in the second-phase of our algorithm. This is an important saving indeed since second-phase subproblems are integer programs.

Step 3.1, Step 5 and Step 6.1. of the algorithm is reshaped with the usage of this subproblem as follows:
Step 3.1. If objective value is larger than p, then set $l = \varepsilon$.
Step 5. If objective value is larger than p, then set set $\varepsilon = u$ else set $\varepsilon = l$.
Step 6.1. If objective value is larger than p, then increase the value of ε:
$\varepsilon' = \min\{d_{ij} : d_{ij} > \varepsilon, \forall i \in V, \forall j \in W\}$ then set $\varepsilon = \varepsilon'$ and go to Step 6.

Daskin's subproblem [3] is a maximal set covering problem, where the number of clients covered within a radius with less than or equal to p facilities is maximized. The formulation of this problem is as follows:

$$\begin{aligned}
&\text{Maximize} && \sum_{i \in V} w_i \\
&\text{subject to} \\
&\sum_{j \in W : d_{ij} \leq \varepsilon} y_j && \geq \quad w_i \qquad \forall i \in V \\
&\sum_{j \in W} y_j \leq p \\
&y_j, w_i && \in \quad \{0,1\} \qquad \forall i \in V,\ \forall j \in W
\end{aligned}$$

where w_i takes value 1 if client i is covered with a facility and 0 otherwise; y_j takes value 1 if a facility is located at site j and 0 otherwise.

With this subproblem Step 3.1, Step 5 and Step 6.1 of the algorithm are given as follows:
Step 3.1. If objective is less than m, then set $l = \varepsilon$.
Step 5. If objective value is less than m, then set $\varepsilon = u$ else set $\varepsilon = l$.
Step 6.1. If objective value is less than m, then increase the value of ε:
$\varepsilon' = \min\{d_{ij} : d_{ij} > \varepsilon, \forall i \in V, \forall j \in W\}$ then set $\varepsilon = \varepsilon'$ and go to Step 6.

In addition to the two subproblems from the literature, a new subproblem, the feasibility subproblem, is formulated and used in the algorithm. This was earlier proposed in a technical report by the first and third authors [6]. In the feasibility subproblem, we aim to find whether it is possible to cover all clients within a coverage distance with at most p facilities. In other words, this subproblem is concerned with whether the following set of inequalities defined with respect to coverage distance ε is feasible.

$$\sum_{j \in W : d_{ij} \leq \varepsilon} y_j \geq 1 \quad \forall i \in V$$
$$\sum_{j} y_j \leq p$$
$$y_j \in \{0,1\} \quad \forall j \in W$$

Step 3.1, Step 5 and Step 6.1 with respect to this subproblem is

Step 3.1. If problem is infeasible, then set $l = \varepsilon$.

Step 5. If problem is infeasible, then set $\varepsilon = u$ else set $\varepsilon = l$.

Step 6.1. If problem is infeasible, then increase the value of ε:
$\varepsilon' = \min \{d_{ij} : d_{ij} > \varepsilon, \forall i \in V, \forall j \in W\}$ then set $\varepsilon = \varepsilon'$ and go to Step 6.

3 Computational Results

In this section we report the computational results obtained with our algorithm for three subproblems. The experiments are carried out on OR-Lib [1] p-median and TSP-Lib [9] problems using CPLEX 7 linear and mixed integer program solvers to solve subproblems. All programs are coded in C [1] and run on Sun UltraSparc Workstation running Solaris.

In OR-Lib, edges and corresponding weights are given for networks of 40 problem instances. We obtained the shortest path distances (d_{ij}'s) by applying Floyd-Warshal algorithm. All shortest path information obtained in this way is integral since edge weights are integral in the data sets. The sizes (i.e. the number of nodes) of the instances vary between 100 and 900.

In TSP-Lib, coordinates of nodes on Euclidean coordinate plane are given. Shortest path distances are computed by finding the Euclidean distances between every pair of nodes. As a result, entries of matrix $\{d_{ij}\}$ are nonintegral. We applied our algorithm to the integral data obtained by rounding every distance to the nearest integer. Also, by making the necessary modification mentioned in section 2.1, we solved TSP-Lib instances for also nonintegral distance matrices. We chose 11 data files and solved each using 5, 10, 20 and 40 as p values. The sizes of instances we chose vary from 200 to 657.

The algorithm is tested on OR-Lib p-median instances, on TSP-Lib problems using integral distances, and on TSP-Lib problems using nonintegral distance matrices. The results of experiments with each subproblem on each problem set are given in nine tables. In the tables, *"file no."* or *"file name"* represents the identification of data file in OR-Lib or TSP-Lib, *"obj"* is the objective function value, and *"cpu t."* the amount of cpu time in seconds. We have calculated total cpu time spent by the algorithm for all of the instances in each table. Deviations from optimal value for bounds obtained in the LP Part are also given. Deviations are calculated by dividing the difference between first and second phase objective function values by optimal value.

Although the deviation from optimality is an important criterion for evaluating the quality of the lower bound obtained from the first phase, another important criterion is the number of iterations after the LP part. There are two reasons for considering this issue. The first one is that the less the number of iterations after the first phase, the less the run time, since the non-polynomial part of our algorithm is the IP part. The second reason is that if the homogeneity of the data is very low, i.e., the difference between distance values is very high, then the first phase of the algorithm

[1] Code is available from the authors upon request.

may give relatively loose lower bounds. Our algorithm works by adding or deleting edges (or arcs) and by considering new radius values in each iteration. In the IP part of the algorithm the radius is changed by increasing it to the next higher distance value. If the difference between consecutive radii is very high, the LP part of the algorithm may fall far from the optimal value even if only one more IP feasibility problem was additionally solved. For instance, consider the problems 1 and 25 in Table 1. In problem 1, seven additional iterations were done over the LP part, and in problem 25, only two. However, the deviations are 4.72% in problem 1, and 9.09% in problem 25. This shows that homogeneity of the problem data is high in the problem 1, but low in problem 25, at least around the optimal values.

It is easy to see that if Minieka's subproblem gives an optimal objective value larger (smaller) than p for a coverage distance ε, then feasibility subproblem must turn out infeasible (feasible) and Daskin's subproblem must yield optimal objective value which is less (larger) than n for ε. In other words, it is impossible to have a coverage distance that turns out to be feasible for the p-center problem with respect to one subproblem and infeasible with respect to another. This fact applies to LP relaxations of the subproblems as well. This means that, irrespective of the subproblem being used, the algorithm makes the same lower and upper bound updates at each iteration of both phases. Thus, number of iterations carried out, bound provided by first phase and its deviation from optimal radius are only dependent on the problem instance being solved, and independent of the subproblem chosen. Due to this fact, the only difference among computational results of the three subproblems arises in the cpu times they spend.

Computational results on OR-Lib problem set show that, in terms of total cpu time, among the three subproblems, the fastest is Minieka's subproblem. Total cpu time spent by the algorithm for all of the 40 OR-Lib instances while working with Minieka's subproblem is 571.49 seconds. This figure turns out to be 639.28 for the feasibility subproblem, and 691.37 seconds for Daskin's.

In TSP-Lib problem set with integral data, we see that total cpu time is again the least for Minieka (1028.04 seconds). Total cpu time spent for these problems with feasibility subproblem is 3141.68 seconds. However, in Table 5 where results for feasibility subproblem is displayed, the cpu time of complete algorithm for problem d657.tsp and p=40 is 2774.98 seconds. If we exclude this problem, the cpu time spent by feasibility subproblem for the rest of the instances becomes 366.7 seconds. The same figure is 851.41 seconds for Minieka's subproblem. This shows that, if we exclude that specific instance, feasibility subproblem is faster than Minieka's by about 8 minutes. The results of Daskin's subproblem for this data set is surprising in the sense that, it could not solve one of the instances within the time limit of 8 hours. Morever, total cpu time of 37588.02 seconds is far worse than the other two, being caused especially by the performance on pr264.tsp and pcb442.tsp for $p = 40$.

When we consider TSP-Lib problems with nonintegral distances, the picture is similar. Minieka's total cpu time is again the least. But if we exclude the same instance (d657.tsp, $p = 40$), from total cpu time, feasibility subproblem is again faster than Minieka's by 1745.27 to 2631.44 seconds. Daskin's subproblem failed to solve the unsolved instance of integral data set, (d657.tsp, $p = 40$). Again the total cpu time even excluding the one that could not be solved in 8 hours is very high (49265.2 seconds) compared to other subproblems.

As a conclusion for cpu times, we can say that feasibility subproblem and Minieka's subproblem are equally fast. But, Minieka's subproblem makes our algorithm most robust.

In the papers by Daskin and Elloumi et al., the computational time results reported are in comparable ranges with the ones we have obtained in terms of cpu time. However, we can not give a precise comparison here since it is not possible for us to make experiments for three algorithms on the same platform.[2]

[2] The algorihtm by Daskin is available for PC's only, while the algorithm by Elloumi et al. requires a Linux CPLEX license.

ORLIB - Subproblem of Minieka									
File			Compl. Algorithm			LP part			
no	size	p	obj.	iter.no.	cpu. t.	obj.	iter.no.	cpu.t.	devia.(%)
1	100	5	127	16	1.34	121	9	0.38	4.72
2	100	10	98	10	0.36	98	9	0.3	0.00
3	100	10	93	11	0.44	92	9	0.33	1.08
4	100	20	74	11	0.31	73	9	0.26	1.35
5	100	33	48	10	0.27	48	9	0.26	0.00
6	200	5	84	10	2.61	83	8	1.55	1.19
7	200	10	64	9	1.42	63	7	1.02	1.56
8	200	20	55	9	1.01	55	8	0.89	0.00
9	200	40	37	10	0.88	36	8	0.73	2.70
10	200	67	20	9	0.72	19	7	0.62	5.00
11	300	5	59	9	4.95	58	7	3.36	1.69
12	300	10	51	10	4.38	50	8	3.17	1.96
13	300	30	36	9	2.53	35	7	1.69	2.78
14	300	60	26	9	1.83	25	7	1.51	3.85
15	300	100	18	9	1.51	17	7	1.33	5.56
16	400	5	47	8	7.48	47	7	6.13	0.00
17	400	10	39	9	10.18	38	7	4.57	2.56
18	400	40	28	9	4.23	27	7	3.17	3.57
19	400	80	18	9	2.45	17	7	2.06	5.56
20	400	133	13	8	2.28	13	7	2.15	0.00
21	500	5	40	8	12.2	40	7	9.34	0.00
22	500	10	38	9	17.57	37	7	9.79	2.63
23	500	50	22	8	5.51	21	6	3.59	4.55
24	500	100	15	9	4.29	14	7	3.74	6.67
25	500	167	11	8	3.86	10	6	3.46	9.09
26	600	5	38	10	30.21	36	7	16.35	5.26
27	600	10	32	8	23.04	31	6	11.97	3.13
28	600	60	18	9	10.59	17	7	7.36	5.56
29	600	120	13	7	5.94	13	6	5.49	0.00
30	600	200	9	8	5.96	9	7	5.73	0.00
31	700	5	30	8	26.28	29	6	16.13	3.33
32	700	10	29	10	64.88	27	7	17.17	6.90
33	700	70	15	8	18.52	14	6	7.69	6.67
34	700	140	11	8	8.69	10	6	7.68	9.09
35	800	5	30	7	36.44	30	6	29.12	0.00
36	800	10	27	8	49.46	27	7	27.39	0.00
37	800	80	15	8	43.54	15	7	11.44	0.00
38	900	5	29	8	59.93	28	6	38.51	3.45
39	900	10	23	8	56.57	22	6	31.5	4.35
40	900	90	13	7	36.83	13	6	13.92	0.00
				Total cpu t.	571.49				

Table 1: Computational results of experiments with Minieka's suproblem on OR-Lib instances

ORLIB - Subproblem of Daskin									
File			Compl. Algorithm			LP part			
name	size	p	obj.	iter.no.	cpu. t.	obj.	iter.no.	cpu.t.	devia.(%)
1	100	5	127	16	1.9	121	9	0.49	4.72
2	100	10	98	10	0.57	98	9	0.49	0.00
3	100	10	93	11	0.56	92	9	0.4	1.08
4	100	20	74	11	0.4	73	9	0.31	1.35
5	100	33	48	10	0.34	48	9	0.3	0.00
6	200	5	84	10	3.91	83	8	2.12	1.19
7	200	10	64	9	2.12	63	7	1.4	1.56
8	200	20	55	9	1.21	55	8	1.02	0.00
9	200	40	37	10	1.16	36	8	0.89	2.70
10	200	67	20	9	0.86	19	7	0.72	5.00
11	300	5	59	9	5.93	58	7	4.26	1.69
12	300	10	51	10	6.3	50	8	4.39	1.96
13	300	30	36	9	4.16	35	7	2.3	2.78
14	300	60	26	9	3.3	25	7	1.9	3.85
15	300	100	18	9	1.85	17	7	1.48	5.56
16	400	5	47	8	8.23	47	7	6.88	0.00
17	400	10	39	9	24.46	38	7	9.16	2.56
18	400	40	28	9	11.73	27	7	3.65	3.57
19	400	80	18	9	5.17	17	7	2.33	5.56
20	400	133	13	8	2.37	13	7	2.2	0.00
21	500	5	40	8	16.1	40	7	14.3	0.00
22	500	10	38	9	21.55	37	7	16.49	2.63
23	500	50	22	8	12.23	21	6	4.57	4.55
24	500	100	15	9	5.83	14	7	4.25	6.67
25	500	167	11	8	4.35	10	6	3.72	9.09
26	600	5	38	10	37.52	36	7	24.23	5.26
27	600	10	32	8	27.69	31	6	20.69	3.13
28	600	60	18	9	23.13	17	7	9.2	5.56
29	600	120	13	7	7.57	13	6	6.04	0.00
30	600	200	9	8	6.81	9	7	6.48	0.00
31	700	5	30	8	38.9	29	6	30.23	3.33
32	700	10	29	10	157.24	27	7	33.03	6.90
33	700	70	15	8	51.06	14	6	9.39	6.67
34	700	140	11	8	12.42	10	6	9.5	9.09
35	800	5	30	7	47.61	30	6	41.29	0.00
36	800	10	27	8	54.13	27	7	47.62	0.00
37	800	80	15	8	23.77	15	7	14.68	0.00
38	900	5	29	8	69.57	28	6	50.91	3.45
39	900	10	23	8	152.52	22	6	52.6	4.35
40	900	90	13	7	30.81	13	6	23.4	0.00
				Total cpu t.	887.34				

Table 2: Computational results of experiments with Daskin's subproblem on OR-Lib instances

ORLIB - Feasibility Subproblem									
File			Compl. Algorithm			LP part			
name	size	p	obj.	iter.no.	cpu. t.	obj.	iter.no.	cpu.t.	devia.(%)
1	100	5	127	16	1.11	121	9	0.37	4.72
2	100	10	98	10	0.43	98	9	0.35	0.00
3	100	10	93	11	0.4	92	9	0.31	1.08
4	100	20	74	11	0.26	73	9	0.21	1.35
5	100	33	48	10	0.27	48	9	0.25	0.00
6	200	5	84	10	2.21	83	8	1.54	1.19
7	200	10	64	9	1.49	63	7	0.95	1.56
8	200	20	55	9	1.02	55	8	0.83	0.00
9	200	40	37	10	0.96	36	8	0.73	2.70
10	200	67	20	9	0.66	19	7	0.55	5.00
11	300	5	59	9	4.82	58	7	3.32	1.69
12	300	10	51	10	4.57	50	8	3.29	1.96
13	300	30	36	9	3.07	35	7	1.63	2.78
14	300	60	26	9	1.97	25	7	1.53	3.85
15	300	100	18	9	1.46	17	7	1.21	5.56
16	400	5	47	8	7.06	47	7	5.61	0.00
17	400	10	39	9	7.25	38	7	4.28	2.56
18	400	40	28	9	4.95	27	7	2.88	3.57
19	400	80	18	9	2.93	17	7	1.9	5.56
20	400	133	13	8	1.99	13	7	1.84	0.00
21	500	5	40	8	10.68	40	7	8.9	0.00
22	500	10	38	9	30.25	37	7	9.52	2.63
23	500	50	22	8	7.92	21	6	3.38	4.55
24	500	100	15	9	4.37	14	7	3.37	6.67
25	500	167	11	8	3.74	10	6	3.24	9.09
26	600	5	38	10	26.65	36	7	15.62	5.26
27	600	10	32	8	19.39	31	6	11.68	3.13
28	600	60	18	9	11.7	17	7	6.79	5.56
29	600	120	13	7	6.74	13	6	5.08	0.00
30	600	200	9	8	5.12	9	7	4.87	0.00
31	700	5	30	8	22.31	29	6	15.05	3.33
32	700	10	29	10	67.2	27	7	16.49	6.90
33	700	70	15	8	11.68	14	6	7.05	6.67
34	700	140	11	8	9.44	10	6	7.2	9.09
35	800	5	30	7	35.37	30	6	27.99	0.00
36	800	10	27	8	45.04	27	7	28.1	0.00
37	800	80	15	8	39	15	7	11.14	0.00
38	900	5	29	8	62.42	28	6	40.34	3.45
39	900	10	23	8	118.21	22	6	31.32	4.35
40	900	90	13	7	53.17	13	6	13.08	0.00
				Total cpu t.	639.28				

Table 3: Computational results of experiments with feasibility subproblem on OR-Lib instances

TSP-LIB(Integral distances) Subproblem of Minieka									
File			Compl. Algorithm			LP part			
name	size	p	obj.	iter.no.	cpu. t.	obj.	iter.no.	cpu.t.	devia(%)
pr226.tsp	226	40	650	16	1.15	649	14	1.03	0.15
pr226.tsp	226	20	1366	19	1.6	1352	14	1.18	1.02
pr226.tsp	226	10	2326	16	1.45	2325	14	1.27	0.04
pr226.tsp	226	5	3721	31	3.98	3658	14	1.62	1.69
pr264.tsp	264	40	316	16	87.72	299	13	1.53	5.38
pr264.tsp	264	20	515	15	1.91	514	13	1.62	0.19
pr264.tsp	264	10	850	15	2.15	849	13	1.85	0.12
pr264.tsp	264	5	1610	14	2.5	1610	13	2.31	0.00
pr299.tsp	299	40	355	15	2.23	354	13	1.95	0.28
pr299.tsp	299	20	559	14	2.45	559	13	2.26	0.00
pr299.tsp	299	10	889	15	4.43	888	13	3.26	0.11
pr299.tsp	299	5	1336	14	4.11	1335	12	3.35	0.07
pr439.tsp	439	40	672	16	7.01	671	14	5.24	0.15
pr439.tsp	439	20	1186	15	6.21	1186	14	5.79	0.00
pr439.tsp	439	10	1972	15	8.19	1971	13	6.92	0.05
pr439.tsp	439	5	3197	15	10.72	3197	14	9.87	0.00
pcb442.tsp	442	40	316	19	473.12	309	13	7.16	2.22
pcb442.tsp	442	20	447	13	10.65	447	12	9.22	0.00
pcb442.tsp	442	10	671	14	11.7	670	12	8.02	0.15
pcb442.tsp	442	5	1025	14	12.09	1024	12	8.74	0.10
kroA200.tsp	200	40	258	14	0.95	257	12	0.86	0.39
kroA200.tsp	200	20	389	13	1.02	389	12	0.95	0.00
kroA200.tsp	200	10	599	14	1.61	598	12	1.22	0.17
kroA200.tsp	200	5	911	15	1.82	910	13	1.52	0.11
kroB200.tsp	200	40	253	14	0.96	252	12	0.84	0.40
kroB200.tsp	200	20	382	17	1.51	378	12	0.96	1.05
kroB200.tsp	200	10	582	14	1.57	581	12	1.29	0.17
kroB200.tsp	200	5	898	13	1.47	898	12	1.35	0.00
lin318.tsp	318	40	316	18	3.47	311	12	1.98	1.58
lin318.tsp	318	20	496	14	2.61	495	12	2.24	0.20
lin318.tsp	318	10	743	14	3.5	743	13	3.24	0.00
lin318.tsp	318	5	1101	13	3.63	1101	12	3.34	0.00
gr202.tsp	202	40	3	8	1.07	3	7	0.96	0.00
gr202.tsp	202	20	6	8	1.17	6	7	1.06	0.00
gr202.tsp	202	10	9	8	1.37	9	7	1.17	0.00
gr202.tsp	202	5	19	9	2.42	18	7	1.74	5.26
d493.tsp	493	40	206	14	13.21	205	12	8.46	0.49
d493.tsp	493	20	313	15	20.36	311	12	11.2	0.64
d493.tsp	493	10	458	14	14.64	457	12	11.98	0.22
d493.tsp	493	5	753	14	16.84	752	12	13.61	0.13
d657.tsp	657	40	250	18	176.63	244	12	18.2	2.40
d657.tsp	657	20	375	14	35.34	374	12	25.41	0.27
d657.tsp	657	10	575	14	41.75	574	12	31.97	0.17
d657.tsp	657	5	881	14	23.75	880	12	19.02	0.11
				Total cpu t.	1028.04				

Table 4: Computational results of experiments with Minieka's subproblem on TSP-Lib instances(integral data)

TSP-LIB(Integral distances) Subproblem of Daskin									
File			Compl. Algorithm			LP part			
no.	size	p	obj.	iter.no.	cpu. t.	obj.	iter.no.	cpu.t.	devia(%)
pr226.tsp	226	40	650	16	1.88	649	14	1.67	0.15
pr226.tsp	226	20	1366	19	2.42	1352	14	1.68	1.02
pr226.tsp	226	10	2326	16	2.24	2325	14	1.95	0.04
pr226.tsp	226	5	3721	31	12.41	3658	14	2.41	1.69
pr264.tsp	264	40	316	16	23990.89	299	13	1.77	5.38
pr264.tsp	264	20	515	15	2.43	514	13	1.99	0.19
pr264.tsp	264	10	850	15	2.36	849	13	2.02	0.12
pr264.tsp	264	5	1610	14	2.76	1610	13	2.54	0.00
pr299.tsp	299	40	355	15	3.67	354	13	3.18	0.28
pr299.tsp	299	20	559	14	4.68	559	13	4.21	0.00
pr299.tsp	299	10	889	15	17.72	888	13	6.45	0.11
pr299.tsp	299	5	1336	14	6.46	1335	12	5.29	0.07
pr439.tsp	439	40	672	16	11.07	671	14	8.29	0.15
pr439.tsp	439	20	1186	15	9.28	1186	14	8.01	0.00
pr439.tsp	439	10	1972	15	11.33	1971	13	9.45	0.05
pr439.tsp	439	5	3197	15	14.5	3197	14	13.48	0.00
pcb442.tsp	442	40	316	19	13129.58	309	13	9.82	2.22
pcb442.tsp	442	20	447	13	12.92	447	12	10.24	0.00
pcb442.tsp	442	10	671	14	31.06	670	12	14.45	0.15
pcb442.tsp	442	5	1025	14	15.49	1024	12	12.96	0.10
kroA200.tsp	200	40	258	14	1.15	257	12	0.99	0.39
kroA200.tsp	200	20	389	13	1.24	389	12	1.14	0.00
kroA200.tsp	200	10	599	14	2.13	598	12	1.63	0.17
kroA200.tsp	200	5	911	15	2.37	910	13	2.03	0.11
kroB200.tsp	200	40	253	14	1.12	252	12	0.97	0.40
kroB200.tsp	200	20	382	17	2.15	378	12	1.23	1.05
kroB200.tsp	200	10	582	14	2.99	581	12	1.73	0.17
kroB200.tsp	200	5	898	13	1.91	898	12	1.76	0.00
lin318.tsp	318	40	316	18	14.97	311	12	2.33	1.58
lin318.tsp	318	20	496	14	3.21	495	12	2.66	0.20
lin318.tsp	318	10	743	14	5.37	743	13	4.97	0.00
lin318.tsp	318	5	1101	13	4.64	1101	12	4.25	0.00
gr202.tsp	202	40	3	8	1.21	3	7	1.1	0.00
gr202.tsp	202	20	6	8	1.32	6	7	1.21	0.00
gr202.tsp	202	10	9	8	1.63	9	7	1.43	0.00
gr202.tsp	202	5	19	9	2.59	18	7	1.09	5.26
d493.tsp	493	40	206	14	15.11	205	12	10.79	0.49
d493.tsp	493	20	313	15	36.09	311	12	17.31	0.64
d493.tsp	493	10	458	14	24.22	457	12	19.19	0.22
d493.tsp	493	5	753	14	19.48	752	12	15.87	0.13
d657.tsp	657	40	Couldn't solve in 8 hours.						
d657.tsp	657	20	375	14	53.7	374	12	32.41	0.27
d657.tsp	657	10	575	14	71.14	574	12	45.87	0.17
d657.tsp	657	5	881	14	33.13	880	12	26.06	0.11
Total cpu t. excluding the one that couldn't be solved					37588.02				

Table 5: Computational results of experiments with Daskin's subproblem on TSP-Lib instances(integral data)

TSP-LIB(Integral distances) Feasibility Subproblem									
File			Compl. Algorithm			LP part			
no.	size	p	obj.	iter.no.	cpu. t.	obj.	iter.no.	cpu.t.	devia(%)
pr226.tsp	226	40	650	16	1.16	649	14	1.04	0.15
pr226.tsp	226	20	1366	19	1.54	1352	14	1.1	1.02
pr226.tsp	226	10	2326	16	1.49	2325	14	1.28	0.04
pr226.tsp	226	5	3721	31	4.26	3658	14	1.52	1.69
pr264.tsp	264	40	316	16	30.94	299	13	1.39	5.38
pr264.tsp	264	20	515	15	1.97	514	13	1.62	0.19
pr264.tsp	264	10	850	15	2	849	13	1.68	0.12
pr264.tsp	264	5	1610	14	2.47	1610	13	2.24	0.00
pr299.tsp	299	40	355	15	2.12	354	13	1.83	0.28
pr299.tsp	299	20	559	14	2.47	559	13	2.18	0.00
pr299.tsp	299	10	889	15	4.76	888	13	2.82	0.11
pr299.tsp	299	5	1336	14	4.11	1335	12	3.3	0.07
pr439.tsp	439	40	672	16	6.65	671	14	4.87	0.15
pr439.tsp	439	20	1186	15	6.21	1186	14	5.41	0.00
pr439.tsp	439	10	1972	15	8.13	1971	13	6.69	0.05
pr439.tsp	439	5	3197	15	10.7	3197	14	9.78	0.00
pcb442.tsp	442	40	316	19	22.41	309	13	6.75	2.22
pcb442.tsp	442	20	447	13	9.69	447	12	7.67	0.00
pcb442.tsp	442	10	671	14	11.55	670	12	7.8	0.15
pcb442.tsp	442	5	1025	14	11.61	1024	12	9.2	0.10
kroA200.tsp	200	40	258	14	0.92	257	12	0.81	0.39
kroA200.tsp	200	20	389	13	1.02	389	12	0.93	0.00
kroA200.tsp	200	10	599	14	1.48	598	12	1.17	0.17
kroA200.tsp	200	5	911	15	1.82	910	13	1.46	0.11
kroB200.tsp	200	40	253	14	0.93	252	12	0.8	0.40
kroB200.tsp	200	20	382	17	1.58	378	12	0.93	1.05
kroB200.tsp	200	10	582	14	1.51	581	12	1.22	0.17
kroB200.tsp	200	5	898	13	1.51	898	12	1.35	0.00
lin318.tsp	318	40	316	18	3.15	311	12	1.81	1.58
lin318.tsp	318	20	496	14	2.51	495	12	2.1	0.20
lin318.tsp	318	10	743	14	3.57	743	13	3.19	0.00
lin318.tsp	318	5	1101	13	3.6	1101	12	3.27	0.00
gr202.tsp	202	40	3	8	1.09	3	7	1	0.00
gr202.tsp	202	20	6	8	1.18	6	7	1.08	0.00
gr202.tsp	202	10	9	8	1.59	9	7	1.26	0.00
gr202.tsp	202	5	19	9	2.48	18	7	1.76	5.26
d493.tsp	493	40	206	14	13.04	205	12	7.9	0.49
d493.tsp	493	20	313	15	38.91	311	12	10.53	0.64
d493.tsp	493	10	458	14	16.52	457	12	12.2	0.22
d493.tsp	493	5	753	14	17.02	752	12	13.85	0.13
d657.tsp	657	40	250	18	2774.98	244	12	13.08	2.40
d657.tsp	657	20	375	14	37.73	374	12	20.08	0.27
d657.tsp	657	10	575	14	34.45	574	12	25.53	0.17
d657.tsp	657	5	881	14	32.85	880	12	20.62	0.11
				Total cpu t.	3141.68				

Table 6: Computational results of experiments with feasibility subproblem on TSP-Lib instances(integral data)

TSP-LIB(Nonintegral distances) Subproblem of Minieka									
File			Compl. Algorithm			LP part			
name	size	p	obj.	iter.no.	cpu. t.	obj.	iter.no.	cpu.t.	devia.(%)
pr226.tsp	226	40	650.00000	15	1.21	650.00000	14	1.15	0.00
pr226.tsp	226	20	1365.65002	19	1.57	1352.08179	14	1.17	0.99
pr226.tsp	226	10	2326.47803	15	1.49	2326.47803	14	1.40	0.00
pr226.tsp	226	5	3720.55103	32	4.34	3658.20996	14	1.75	1.68
pr264.tsp	264	40	316.22775	15	89.84	300.00000	13	1.66	5.13
pr264.tsp	264	20	514.78149	14	1.85	514.78149	13	1.71	0.00
pr264.tsp	264	10	850.00000	14	2.13	850.00000	13	1.98	0.00
pr264.tsp	264	5	1610.12427	14	2.70	1610.12427	13	2.51	0.00
pr299.tsp	299	40	355.31677	15	2.34	355.00140	13	2.07	0.09
pr299.tsp	299	20	559.01697	14	2.60	559.01697	13	2.43	0.00
pr299.tsp	299	10	888.83575	17	5.31	888.32428	13	3.30	0.06
pr299.tsp	299	5	1336.27283	15	4.62	1336.00000	13	3.84	0.02
pr439.tsp	439	40	671.75146	15	6.46	671.75146	14	5.59	0.00
pr439.tsp	439	20	1185.59058	16	7.01	1185.32690	14	6.13	0.02
pr439.tsp	439	10	1971.83289	16	9.43	1971.19885	13	7.40	0.03
pr439.tsp	439	5	3196.58032	15	11.47	3196.58032	14	10.60	0.00
pcb442.tsp	442	40	316.22775	23	886.86	310.00000	13	7.61	1.97
pcb442.tsp	442	20	447.21359	13	10.73	447.21359	12	9.06	0.00
pcb442.tsp	442	10	670.82037	16	26.68	670.00000	12	8.82	0.12
pcb442.tsp	442	5	1024.74390	16	17.28	1024.15820	12	9.25	0.06
kroA200.tsp	200	40	258.25955	16	1.08	257.49756	12	0.87	0.30
kroA200.tsp	200	20	389.30707	13	1.14	389.30707	12	1.07	0.00
kroA200.tsp	200	10	598.81970	15	1.94	598.36194	12	1.38	0.08
kroA200.tsp	200	5	911.41211	14	1.77	911.41211	13	1.64	0.00
kroB200.tsp	200	40	253.23705	16	1.11	253.02371	12	0.90	0.08
kroB200.tsp	200	20	382.28000	38	4.39	377.07956	12	1.08	1.36
kroB200.tsp	200	10	582.10394	15	1.99	582.00342	12	1.42	0.02
kroB200.tsp	200	5	897.66919	19	2.40	897.11761	12	1.44	0.06
lin318.out	318	40	315.91928	25	4.72	312.06570	13	2.21	1.22
lin318.out	318	20	496.45242	14	3.27	496.13910	12	2.38	0.06
lin318.out	318	10	743.21063	13	3.42	743.21063	12	3.14	0.00
lin318.out	318	5	1101.33960	14	4.36	1101.11804	12	3.63	0.02
gr202.tsp	202	40	2.97136	446	41.01	2.04012	7	1.07	31.34
gr202.tsp	202	20	5.56569	439	55.94	5.04001	7	1.18	9.45
gr202.tsp	202	10	9.33400	322	67.62	9.04067	7	1.34	3.14
gr202.tsp	202	5	19.38451	226	78.57	19.04124	7	1.89	1.77
d493.tsp	493	40	206.01566	15	13.90	205.96585	12	9.60	0.02
d493.tsp	493	20	312.74490	69	319.05	311.98709	12	12.56	0.24
d493.tsp	493	10	458.30457	40	68.53	457.96622	12	12.69	0.07
d493.tsp	493	5	752.90845	62	110.65	751.97314	12	15.00	0.12
d657.tsp	657	40	249.51541	64	1686.84	244.28938	12	18.60	2.09
d657.tsp	657	20	374.70001	22	224.32	373.96548	12	24.33	0.20
d657.tsp	657	10	574.74469	31	205.29	574.16077	12	28.45	0.10
d657.tsp	657	5	880.90851	48	319.05	879.97345	12	20.69	0.11
				Total cpu t.	4318.28				

Table 7: Computational results of experiments with Minieka's subproblem on TSP-Lib instances(nonintegral data)

TSP-LIB(Nonintegral distances) Subproblem of Daskin									
File			Compl. Algorithm			LP part			
name	size	p	obj.	iter.no.	cpu. t.	obj.	iter.no.	cpu.t.	devia.(%)
pr226.tsp	226	40	650.00000	15	1.38	650.00000	14	1.31	0.00
pr226.tsp	226	20	1365.65002	19	1.97	1352.08179	14	1.40	0.99
pr226.tsp	226	10	2326.47803	15	1.67	2326.47803	14	1.56	0.00
pr226.tsp	226	5	3720.55103	32	10.09	3658.20996	14	1.86	1.68
pr264.tsp	264	40	316.22775	15	23169.23	300.00000	13	1.64	5.13
pr264.tsp	264	20	514.78149	14	2.25	514.78149	13	2.02	0.00
pr264.tsp	264	10	850.00000	14	2.23	850.00000	13	2.07	0.00
pr264.tsp	264	5	1610.12427	14	2.84	1610.12427	13	2.61	0.00
pr299.tsp	299	40	355.31677	15	2.80	355.00140	13	2.40	0.09
pr299.tsp	299	20	559.01697	14	3.47	559.01697	13	3.09	0.00
pr299.tsp	299	10	888.83575	17	18.83	888.32428	13	4.11	0.06
pr299.tsp	299	5	1336.27283	15	5.22	1336.00000	13	4.29	0.02
pr439.tsp	439	40	671.75146	15	7.99	671.75146	14	6.82	0.00
pr439.tsp	439	20	1185.59058	16	8.61	1185.32690	14	6.74	0.02
pr439.tsp	439	10	1971.83289	16	10.31	1971.19885	13	7.95	0.03
pr439.tsp	439	5	3196.58032	15	12.20	3196.58032	14	11.29	0.00
pcb442.tsp	442	40	316.22775	23	23661.75	310.00000	13	9.39	1.97
pcb442.tsp	442	20	447.21359	13	12.80	447.21359	12	11.23	0.00
pcb442.tsp	442	10	670.82037	16	40.80	670.00000	12	14.60	0.12
pcb442.tsp	442	5	1024.74390	16	18.45	1024.15820	12	12.79	0.06
kroA200.tsp	200	40	258.25955	16	1.29	257.49756	12	0.98	0.30
kroA200.tsp	200	20	389.30707	13	1.33	389.30707	12	1.23	0.00
kroA200.tsp	200	10	598.81970	15	2.37	598.36194	12	1.62	0.08
kroA200.tsp	200	5	911.41211	14	2.17	911.41211	13	2.03	0.00
kroB200.tsp	200	40	253.23705	16	1.29	253.02371	12	0.97	0.08
kroB200.tsp	200	20	382.28000	38	10.58	377.07956	12	1.26	1.36
kroB200.tsp	200	10	582.10394	15	2.62	582.00342	12	1.76	0.02
kroB200.tsp	200	5	897.66919	19	2.87	897.11761	12	1.75	0.06
lin318.tsp	318	40	315.91928	25	30.51	312.06570	13	2.59	1.22
lin318.tsp	318	20	496.45242	14	4.45	496.13910	12	2.79	0.06
lin318.tsp	318	10	743.21063	13	5.00	743.21063	12	4.63	0.00
lin318.tsp	318	5	1101.33960	14	5.52	1101.11804	12	4.65	0.02
gr202.tsp	202	40	2.97136	446	83.18	2.04012	7	1.07	31.34
gr202.tsp	202	20	5.56569	439	113.22	5.04001	7	1.24	9.45
gr202.tsp	202	10	9.33400	322	95.19	9.04067	7	1.40	3.14
gr202.tsp	202	5	19.38451	226	84.54	19.04124	7	1.92	1.77
d493.tsp	493	40	206.01566	15	42.48	205.96585	12	10.80	0.02
d493.tsp	493	20	312.74490	69	838.87	311.98709	12	16.99	0.24
d493.tsp	493	10	458.30457	40	115.51	457.96622	12	17.12	0.07
d493.tsp	493	5	752.90845	62	128.80	751.97314	12	15.14	0.12
d657.tsp	657	40	Couldn't solve in 8 hours.						
d657.tsp	657	20	374.70001	22	217.12	373.96548	12	29.48	0.20
d657.tsp	657	10	574.74469	31	176.35	574.16077	12	40.33	0.10
d657.tsp	657	5	880.90851	48	417.93	879.97345	12	33.06	0.11
Total cpu t. excluding the one that couldn't be solved.					49265.2				

Table 8: Computatinal results of experiments with Daskin's subproblem on TSP-Lib instances(nonintegral data)

TSP-LIB(Nonintegral distances) Feasibility Subproblem									
File			Compl. Algorithm			LP part			
name	size	p	obj.	iter.no.	cpu. t.	obj.	iter.no.	cpu.t.	devia(%)
pr226.tsp	226	40	650.00000	15	1.12	650.00000	14	1.07	0.00
pr226.tsp	226	20	1365.65002	19	1.67	1352.08179	14	1.17	0.99
pr226.tsp	226	10	2326.47803	15	1.46	2326.47803	14	1.35	0.00
pr226.tsp	226	5	3720.55103	32	4.74	3658.20996	14	1.71	1.68
pr264.tsp	264	40	316.22775	15	30.85	300.00000	13	1.45	5.13
pr264.tsp	264	20	514.78149	14	1.86	514.78149	13	1.68	0.00
pr264.tsp	264	10	850.00000	14	2.02	850.00000	13	1.87	0.00
pr264.tsp	264	5	1610.12427	14	2.59	1610.12427	13	2.35	0.00
pr299.tsp	299	40	355.31677	15	2.23	355.00140	13	1.94	0.09
pr299.tsp	299	20	559.01697	14	2.57	559.01697	13	2.28	0.00
pr299.tsp	299	10	888.83575	17	5.61	888.32428	13	2.98	0.06
pr299.tsp	299	5	1336.27283	15	4.42	1336.00000	13	3.65	0.02
pr439.tsp	439	40	671.75146	15	6.60	671.75146	14	5.37	0.00
pr439.tsp	439	20	1185.59058	16	7.33	1185.32690	14	5.78	0.02
pr439.tsp	439	10	1971.83289	16	9.59	1971.19885	13	7.35	0.03
pr439.tsp	439	5	3196.58032	15	11.58	3196.58032	14	10.60	0.00
pcb442.tsp	442	40	316.22775	23	36.88	310.00000	13	7.10	1.97
pcb442.tsp	442	20	447.21359	13	9.62	447.21359	12	7.87	0.00
pcb442.tsp	442	10	670.82037	16	16.52	670.00000	12	8.40	0.12
pcb442.tsp	442	5	1024.74390	16	14.74	1024.15820	12	9.81	0.06
kroA200.tsp	200	40	258.25955	16	1.11	257.49756	12	0.86	0.30
kroA200.tsp	200	20	389.30707	13	1.10	389.30707	12	1.00	0.00
kroA200.tsp	200	10	598.81970	15	1.67	598.36194	12	1.25	0.08
kroA200.tsp	200	5	911.41211	14	1.76	911.41211	13	1.58	0.00
kroB200.tsp	200	40	253.23705	16	1.11	253.02371	12	0.85	0.08
kroB200.tsp	200	20	382.28000	38	4.23	377.07956	12	0.98	1.36
kroB200.tsp	200	10	582.10394	15	1.74	582.00342	12	1.30	0.02
kroB200.tsp	200	5	897.66919	19	2.36	897.11761	12	1.33	0.06
lin318.tsp	318	40	315.91928	25	5.21	312.06570	13	2.00	1.22
lin318.tsp	318	20	496.45242	14	2.64	496.13910	12	2.20	0.06
lin318.tsp	318	10	743.21063	13	3.46	743.21063	12	3.04	0.00
lin318.tsp	318	5	1101.33960	14	4.28	1101.11804	12	3.53	0.02
gr202.tsp	202	40	2.97136	446	33.31	2.04012	7	1.07	31.34
gr202.tsp	202	20	5.56569	439	64.39	5.04001	7	1.08	9.45
gr202.tsp	202	10	9.33400	322	72.19	9.04067	7	1.26	3.14
gr202.tsp	202	5	19.38451	226	84.70	19.04124	7	1.79	1.77
d493.tsp	493	40	206.01566	15	14.15	205.96585	12	8.53	0.02
d493.tsp	493	20	312.74490	69	652.97	311.98709	12	10.93	0.24
d493.tsp	493	10	458.30457	40	74.71	457.96622	12	12.39	0.07
d493.tsp	493	5	752.90845	62	99.18	751.97314	12	14.62	0.12
d657.tsp	657	40	249.51541	64	9333.92	244.28938	12	13.30	2.09
d657.tsp	657	20	374.70001	22	65.00	373.96548	12	21.91	0.20
d657.tsp	657	10	574.74469	31	119.26	574.16077	12	25.98	0.10
d657.tsp	657	5	880.90851	48	258.74	879.97345	12	25.44	0.11
				Total cpu t.	11079.19				

Table 9: Computational results of experimens with feasibility subproblem on TSP-Lib instances(nonintegral data)

References

[1] J.E. Beasley (1985), A note on solving large p-median problems, *European J. Oper. Res.* 21, 270–273.

[2] M. Daskin (1995). *Network and Discrete Location*, Wiley, New York.

[3] M. Daskin (2000), A new approach to solving the certex p-center problem to optimality: algorithm and computational results. *Communications of the Operations Research Society of Japan* 45:9, 428–436.

[4] S. Elloumi, M. Labbé, Y. Pochet (2001), New Formulation and Resolution Method for the p-Center Problem, *http://www.optimization-online.org/DB-HTML/2001/10/394.html*

[5] O. Kariv and S.L. Hakimi (1979), An algorithmic approach to network location problems Part I: The p-centers, *SIAM J. Appl. Math.* 37, 513–538.

[6] T. Ilhan, M. Ç. Pınar (2001), An Efficient Exact Algorithm for the Vertex p-Center Problem, http://www.optimization-online.org/DB-HTML/2001/09/376.html.

[7] E. Minieka (1970), The m-center problem, *SIAM Review* 12, 138–139.

[8] N. Mladenovic, M. Labbé, P. Hansen (2000), Solving the p-center problem with tabu search and variable neighborhood search, Technical Report, SMG, Université Libre de Bruxelles, available from `smg.ulb.ac.be/Preprints/Labbe00_20.html`.

[9] G. Reinelt (1991), TSP-Lib-A traveling salesman problem library, *ORSA J. Comput.* 3, 376–384.