An Efficient Exact Algorithm for the Vertex p-Center Problem

Taylan Ilhan*
Mustafa Ç. Pınar †

September 27, 2001

Abstract

Inspired by an algorithm due to E. Minieka [5] 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. After a lower bounding phase, the algorithm iteratively sets a maximum distance value within which it tries to assign all clients, and thus solves integer feasibility subproblems. Excellent computational results are reported on a set of 84 test problems derived from OR-Lib and TSP-Lib problem instances with up to 900 vertices, and solved to optimality for the first time.

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

1 Introduction

The purpose of this paper is to describe a simple and efficient algorithm for the solution of the vertex (discrete) p-center problem. The algorithm can be considered a two-phase extension of a well-known algorithm due to Minieka [5] long considered inefficient. In fact, in a recent paper where tabu search and variable neighborhood search heuristics are applied to the p-center problem [4], the authors claim that "no exact algorithm able to solve large instances (of the p-center problem) seems to exist up to now". The computational results of this paper serve as an update to this statement by solving to provable optimality large scale p-center problems for the first time, to the best of the authors' knowledge.

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 [3]. 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].

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

*E-mail: *taylani@bilkent.edu.tr*

†Corresponding author. E-mail: *mustafap@bilkent.edu.tr*. Both authors are at the Department of Industrial Engineering, Bilkent University, 06533 Ankara, Turkey.

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 that we refer to as (PCIP) is the following (see, e.g., [2]):

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

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). It is well-known that solving to optimality even small instances ($N + M = 40, p = 10$) of the p-center problem using the PCIP formulation is still very time consuming (more than 7 hours on a SUN Sparc 10 using CPLEX 6.0 [4]), and that LP relaxations to the above PCIP formulation give notoriously loose lower bounds, an observation reconfirmed in our own experimentation, whereas none of these statements are true of the p-median problem where the sum of total distances are minimized.

In Section 2 we explain the idea underlying the proposed algorithm and describe it in detail. In Section 3 we give our detailed computational results.

2 The Algorithm

An old algorithm due to Minieka [5] proposes to solve the p-center problem by solving a series of set covering problems. The idea is to choose a threshold distance as radius and to check whether all clients can be covered within this radius using no more than p facilities. A more elaborate version of this algorithm as described in [2] works as follows: select initial lower and upper bounds on the value of the p-center objective function, solve a set covering problem using the average of the lower and upper bounds on the objective function as the coverage distance. If the number of facilities needed to cover all nodes at that distance is less than or equal to p, reset the value of the upper bound on the value of the p-center objective function to the coverage distance that was just used; if the number of facilities needed is greater than p, reset the lower bound to the coverage distance that was just used plus 1. If the lower and upper bounds are equal, stop; if not, solve the set covering problem with a coverage distance equal to the average of the lower and upper bounds and continue the process.

Our algorithm can be considered a two-phase extension of this idea. In the first phase simple LP feasibility problems with covering and upper bound on the sum of open facilities constraints are solved to obtain a suitable lower bound on the optimal value. In the second phase, starting from this lower bound a series of IP feasibility problems of the same type as above (as to whether there is a solution to the problem with $z = \varepsilon$ or not), with different ε values for the radius are solved. The feasibility problem for a specific radius ε, that we refer to as (IP), is given below:

$$\sum_{j \in W} \bar{b}_{ij} w_j \geq 1 \quad \forall i \in V \tag{2.1}$$

$$\sum_{j \in W} w_j \leq p \tag{2.2}$$

$$w_j \in \{0, 1\} \quad \forall j \in W \tag{2.3}$$

where w_j is one if facility site j is open, and, $\bar{b}_{ij}$ is a parameter calculated as:

$$\bar{b}_{ij} = \begin{cases} 1 & d_{ij} \leq \varepsilon \\ 0 & \text{otherwise.} \end{cases}$$

If this (IP) formulation is infeasible, it means that for the specific radius ε, there is no feasible solution to p-center problem and ε is less than the optimal solution of the problem. Thus we have to increase the maximum radius, ε, until the (IP) formulation above becomes feasible. At the ε value that we just reach the feasibility, we also reach optimality in 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 $\varepsilon = \lceil (u - l)/2 \rceil$
Step 2.1. if $d_{ij} \leq \varepsilon$ then set $b_{ij} = 1$
Step 2.2. else set $b_{ij} = 0$
Step 3. Solve the LP relaxation of (IP) above by replacing constraint (2.3) with:

$$0 \leq w_j \leq 1 \quad \forall j \in W \tag{2.4}$$

Step 3.1. If the LP relaxation is not feasible 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 the LP formulation is not feasible then Set $\varepsilon = u$ else set $\varepsilon = l$
Step 6. If $d_{ij} \leq \varepsilon$ then set $b_{ij} = 1$ else set $b_{ij} = 0$
Step 7. Solve the (IP) problem.
Step 7.1. If (IP) 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.
Step 7.2. else (IP feasible) Stop.

In our algorithm, we solve LP and IP formulations iteratively by changing the radius of the p-center problem. In the first part of the algorithm (Step 1 through Step 5), where we solve LP problems, we search for the best possible bound that will be used as the starting radius for the IP part(Steps 6 and 7). In the LP part, we use the bisection method to find the minimum radius that maximize the objective function. In the second part where we solve IP feasibility problems, we start from the minimum ε value found by the first part of the algorithm. Until reaching a feasible solution, we continue to solve IP formulations by increasing ε to the minimum distance value that is greater than ε. When we reach a feasible solution to (IP), we can conclude that we obtained an optimal solution to our p-center problem.

[1]Although this algorithm is for integer data set, it can be easily modified for real data sets.

3 Computational Results

In this section, we report the computational results obtained with our algorithm on OR-Lib [1] p-median and TSP-Lib [6] problems using CPLEX 5 linear and mixed-integer program solvers[1]. We note that these problems are solved to optimality for the first time in the literature, to the best of our knowledge. All programs[2] are coded in C and tested on a Sun Enterprise Workstation 4000 running Solaris 7. We use the CPLEX 5 LP and MIP solvers to solve the LP and IP feasibility problems in our algorithm. Since there is no specific algorithm in the literature available to compare with our algorithm for large scale problems, we used the CPLEX 5 MIP solver on the PCIP formulation to compare with our algorithm with respect to time efficiency. The comparison is limited to the first ten OR-Lib p-median problems as the solution of the p-center PCIP formulation consumes very large amounts of computer time already for small problems.

The results of the comparison on p-median problems are given in Table 1 where "size" refers to the number of vertices in the graph. Since the size of the problems are very large for the PCIP formulation we put a time limit of 3600 seconds. This time limit is very reasonable when we compared with the run times of our algorithm. In our experiments we also compared the results of LP relaxation of the PCIP formulation with the results obtained from LP phase of our algorithm. *"File no."* represents the file number of p-median data in OR-Lib, *"obj."* is the objective function value, and *"cpu t."* is the amount of cpu time in seconds.

			PCIP Formulation				**P**roposed Algorithm			
			LP Relaxation		**P**CIP		**L**P Part		**L**P+IP	
file no.	size	p	obj.	cpu t.	obj.	cpu t.	obj.	cpu t.	obj.	cpu t.
1	100	5	90.92	60.98	127	3601.79	121	0.89	127	2.08
2	100	10	63.35	49.21	110	3601.74	98	0.78	98	0.87
3	100	10	62.48	82.98	137	3601.44	92	0.69	93	0.83
4	100	20	41.5	58.66	119	3601.41	73	0.52	74	0.64
5	100	33	19.12	49.41	60	3601.67	48	0.53	48	0.58
6	200	5	62.87	2017.63	150	3611.58	83	3.17	84	6.13
7	200	15	37.14	1654.81	90	3607.54	55	2.03	56	2.7
8	200	20	33.84	1647.99	202	3605.25	55	1.66	55	1.88
9	200	40	20.02	747.19	65	3609.69	36	1.51	37	1.74
10	200	67	8.76	524.76	75	3605.49	19	1.2	20	1.37

Table 1: Comparison of PCIP Formulation on CPLEX 5 with Proposed Algorithm over p-median data

As can be seen in the table, the PCIP formulation of all of the problems reaches the time limit without finding the optimal solution. In all cases, we report the best integer solution value found by CPLEX when the 1 hour cpu time limit was reached. Although for the first problem instance it reached the optimal solution, it continued to run to be able to verify optimality of the solution. However, if we look at the cpu times of our algorithm, it found optimal solutions for each problem instance in less than 7 seconds, and in 1.9 seconds on average. This shows that our algorithm is computationally far more efficient than the PCIP formulation on a general purpose MIP solver.

We also tested the first phase of our algorithm and LP relaxation of the PCIP formulation. When we look at the computational efficiency and the quality of the lower bound obtained, the first phase of our algorithm gives lower bounds much superior to the regular LP relaxation. The average run times for LP relaxation and LP part of our algorithm are 689.36 and 1.30 seconds, respectively. The deviations from the optimal values are also given in the Table 2. These deviations are calculated by taking the distance between the objective function value and the optimal value and then dividing it by the optimal value.

[1]We do not have access to higher versions.
[2]The code is available from the authors upon request.

file no.	PCIP (%)	LP relax. (%)	LP part (%)
1	0.00	28.41	4.72
2	12.24	35.36	0.00
3	47.31	32.82	1.08
4	60.81	43.92	1.35
5	25.00	60.17	0.00
6	78.57	25.15	1.19
7	60.71	33.68	1.79
8	267.27	38.47	0.00
9	75.68	45.89	2.70
10	275.00	56.20	5.00
Average	90.26	40.01	1.78
Max	275.00	60.17	5.00

Table 2: Deviations of PCIP Formulation, its LP relaxation, and LP part of the algorithm from the optimal solution for each p-median problem instance

In the first column of Table 2, it is observed that the deviation from the optimal value for the PCIP formulation goes up to 275% in spite of the 3600 seconds run time. In the second and the third columns, the obvious superiority of our algorithm appears. The regular LP relaxation deviates up to 60%, but our algorithm's LP part deviates at most by 5%, and it finds the optimal value three times. However we should state that finding the optimal value does not necessarily mean finding an optimal integer solution.

The results of the algorithm for all OR-Lib problems are given in Table 3. In the table, two new columns named *"iter. no."*, which means number of iterations, are included. The first one shows how many feasibility problems, both LP and IP, have been solved until reaching the optimal solution. The second one shows how many LP relaxations of the feasibility problem mentioned in the Section 2 are solved or, in other words, how many iterations have been done until reaching step 5 of the algorithm.

The run times of the algorithm are also very low when one considers the sizes of the problems. Even for the problems with 900 vertices, run times are less than 10 minutes of cpu time. Also the quality of lower bounds obtained from the first phase of the algorithm is very high. The deviations from the optimal value never goes beyond 10%, and for 12 out of 40, it finds the optimal values.

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 done 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 are very high, then the first phase of the algorithm 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 radiuses 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 3. In problem 1, six 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.

File			Compl. Algorithm			LP Part			
no.	size	p	obj.	iter no.	cpu t.	obj.	iter no.	cpu t.	devia.(%)
1	100	5	127	16	2.08	121	9	0.89	4.72
2	100	10	98	10	0.87	98	9	0.78	0.00
3	100	10	93	11	0.83	92	9	0.69	1.08
4	100	20	74	11	0.64	73	9	0.52	1.35
5	100	33	48	10	0.58	48	9	0.53	0.00
6	200	5	84	10	6.13	83	8	3.17	1.19
7	200	15	56	10	2.7	55	8	2.03	1.79
8	200	20	55	9	1.88	55	8	1.66	0.00
9	200	40	37	10	1.74	36	8	1.51	2.70
10	200	67	20	9	1.37	19	7	1.2	5.00
11	300	5	59	9	9.13	58	7	6.85	1.69
12	300	10	51	10	8.21	50	8	6.5	1.96
13	300	30	36	9	4.2	35	7	3.27	2.78
14	300	60	26	9	3.42	25	7	2.99	3.85
15	300	100	18	9	2.72	17	7	2.38	5.56
16	400	5	47	8	13.94	47	7	11.93	0.00
17	400	10	39	9	13.41	38	7	9.05	2.56
18	400	40	28	9	19.42	27	7	5.92	3.57
19	400	80	18	9	4.85	17	7	3.85	5.56
20	400	133	13	8	4.08	13	7	3.86	0.00
21	500	5	40	8	42.34	40	7	19.79	0.00
22	500	10	38	9	130.46	37	7	20	2.63
23	500	50	22	8	35.81	21	6	7.13	4.55
24	500	100	15	9	7.84	14	7	7	6.67
25	500	167	11	8	7.06	10	6	6.35	9.09
26	600	5	38	10	121.72	36	7	34.09	5.26
27	600	10	32	8	73.53	31	6	25.11	3.13
28	600	60	18	9	18.16	17	7	13.48	5.56
29	600	120	13	7	10.18	13	6	9.67	0.00
30	600	200	9	8	9.99	9	7	9.59	0.00
31	700	5	30	8	108.22	29	6	33	3.33
32	700	10	29	10	460.34	27	7	33.46	6.90
33	700	70	15	8	32.37	14	6	14.51	6.67
34	700	140	11	8	15.56	10	6	14.13	9.09
35	800	5	30	7	66.46	30	6	56.39	0.00
36	800	10	27	8	342.1	27	7	55.53	0.00
37	800	80	15	8	35.18	15	7	33.2	0.00
38	900	5	29	8	96.04	28	6	68.05	3.45
39	900	10	23	8	536.48	22	6	57.38	4.35
40	900	90	13	7	404.9	13	6	28.94	0.00

Table 3: Results of the experiments with p-median data

File no.	size	p	Compl. Algorithm obj.	iter no.	cpu t.	LP part obj.	iter no.	cpu t.	devia. (%)
pr226.tsp	226	40	650	16	2.24	649	14	2.02	0.15
pr226.tsp	226	20	1366	19	2.9	1352	14	2.21	1.02
pr226.tsp	226	10	2326	16	2.8	2325	14	2.5	0.04
pr226.tsp	226	5	3721	31	7.77	3658	14	3.13	1.69
pr264.tsp	264	40	316	16	130.97	299	13	2.63	5.38
pr264.tsp	264	20	515	15	3.54	514	13	3.11	0.19
pr264.tsp	264	10	850	15	3.91	849	13	3.44	0.12
pr264.tsp	264	5	1610	14	4.74	1610	13	4.42	0.00
pr299.tsp	299	40	355	15	4.01	354	13	3.67	0.28
pr299.tsp	299	20	559	14	4.71	559	13	4.49	0.00
pr299.tsp	299	10	889	15	9.59	888	13	6.65	0.11
pr299.tsp	299	5	1336	14	8.38	1335	12	7.16	0.07
pr439.tsp	439	40	672	16	12.36	671	14	10.28	0.15
pr439.tsp	439	20	1186	15	18.99	1186	14	15.7	0.00
pr439.tsp	439	10	1972	15	17.32	1971	13	14.53	0.05
pr439.tsp	439	5	3197	15	29.64	3197	14	27.49	0.00
pcb442.tsp	442	40	316	19	39.72	309	13	13.77	2.22
pcb442.tsp	442	20	447	13	38.9	447	12	16.9	0.00
pcb442.tsp	442	10	671	14	53.65	670	12	27.28	0.15
pcb442.tsp	442	5	1025	14	107.31	1024	12	82.01	0.10
kroA200.tsp	200	40	258	14	1.76	257	12	1.58	0.39
kroA200.tsp	200	20	389	13	1.91	389	12	1.8	0.00
kroA200.tsp	200	10	599	14	2.91	598	12	2.27	0.17
kroA200.tsp	200	5	911	15	3.48	910	13	3.04	0.11
kroB200.tsp	200	40	253	14	1.75	252	12	1.55	0.40
kroB200.tsp	200	20	382	17	2.71	378	12	1.84	1.05
kroB200.tsp	200	10	582	14	2.67	581	12	2.28	0.17
kroB200.tsp	200	5	898	13	3.38	898	12	3.07	0.00
lin318.tsp	318	40	316	18	5.3	311	12	3.67	1.58
lin318.tsp	318	20	496	14	4.76	495	12	4.23	0.20
lin318.tsp	318	10	743	14	7.64	743	13	7.22	0.00
lin318.tsp	318	5	1101	13	9.14	1101	12	8.68	0.00
gr202.tsp	202	40	3	8	1.96	3	7	1.86	0.00
gr202.tsp	202	20	6	8	2.44	6	7	2.25	0.00
gr202.tsp	202	10	9	8	2.98	9	7	2.73	0.00
gr202.tsp	202	5	19	9	4.41	18	7	3.35	5.26
d493.tsp	493	40	206	14	36.03	205	12	27.18	0.49
d493.tsp	493	20	313	15	106.89	311	12	71.81	0.64
d493.tsp	493	10	458	14	81.7	457	12	67	0.22
d493.tsp	493	5	753	14	78.48	752	12	56.09	0.13
d657.tsp	657	40	250	18	7072.22	244	12	26.94	2.40
d657.tsp	657	20	375	14	154.39	374	12	60.67	0.27
d657.tsp	657	10	575	14	196.74	574	12	166.55	0.17
d657.tsp	657	5	881	14	360.28	880	12	303.53	0.11

Table 4: Results of the experiments with TSP data

However, even though the deviations in some problems, such as problem 25 and 33, are higher than some other deviation values, the number of iterations of the IP part still remains quite low. Therefore, when we consider the number of iterations, we can accept the objective function values produced by the first phase as good lower bounds if the second phase uses a small number of iterations.

In addition to p-median problems from OR-Lib data set, we also applied our algorithm to some problems from the TSP-LIB data set. The sizes of the problems range from 200 nodes to 657 nodes. We chose four different p values for each problem as 5, 10, 20, and 40. The results are given in Table 4. If we consider the results of the LP part, the maximum deviation from the optimal value is 5.28% and average run time is 24.65 seconds. The complete algorithm is also computationally very efficient with an average 196.58 seconds. For one case, problem "d657.tsp", the cpu time is relatively high, 7072 seconds. However, it is still very good if we consider the PCIP formulation where we can safely claim that it cannot be solved for the same problem in a reasonable amount of computer time.

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

[4] 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`.

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

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