REVISITING RELAXATION-BASED OPTIMAL ALGORITHMS FOR THE SOLUTION OF THE CONTINUOUS AND DISCRETE P-CENTER PROBLEMS

Doron Chen[1] and Reuven Chen[2]

ABSTRACT. We present a new variant of relaxation algorithms for the continuous and discrete p-center problems. We have conducted an experimental study that demonstrated that these new variants are very efficient, and often outperform other optimal algorithms.

Keywords: P-center, Relaxation

1. INTRODUCTION

The p-center problem (see, for example, [13]), also known as the minimax location-allocation problem, deals with the optimal location of emergency facilities. The locations of n demand points are given, and the objective is to locate p service facilities so as to minimize the maximum distance between a demand point to its nearest service facility. It is assumed that all the facilities perform the same kind of service, and that the number of demand points that can get service from a given center is unlimited.

Relaxation (in the context of this paper) [13, 5] is a simple method to *optimally* solve a large location problem by solving a succession of small sub-problems. Fortunately, optimality can be achieved even though each sub-problem need not be solved to optimality. Relaxation algorithms are iterative; at each step a candidate solution to the original full problem is considered, which provides us with an upper bound on the optimal solution. We search, at each step, for a solution which improves upon the current candidate solution, until we prove that none exists.

Algorithm 1 describes the skeleton of a relaxation algorithm.

When Algorithm 1 halts, $Best\_Candidate$ contains a solution to the original (full) location problem with value $Upper\_Bound$. Furthermore, the algorithm halts after FINDFEASIBLESOLUTION fails to find a feasible solution with value better than

[1]Doron Chen, IBM Haifa Research Lab, Tel-Aviv Site. cdoron@il.ibm.com

[2]Reuven Chen, School of Physics and Astronomy, Raymond and Beverly Sackler Faculty of Exact Sciences, Tel-Aviv University, Tel-Aviv 69978, Israel. chenr@tau.ac.il

Algorithm 1 Skeleton of Relaxation Algorithms.

```
$Upper\_Bound \leftarrow \infty$
$Sub \leftarrow$ CHOOSERANDOMSUBSET()
while (solution not found)
  $Feasible \leftarrow$FINDFEASIBLESOLUTION($Sub$, $Upper\_Bound$)
  if (no feasible solution found for sub-problem)
    halt and return $Best\_Candidate$
  else
    if ($Feasible$ is a feasible solution to the original full problem)
      $Best\_Candidate \leftarrow Feasible$
      $Upper\_Bound \leftarrow$GETVALUE($Feasible$)
    else
      ADDDEMANDPOINT($Sub$)
```

$Upper\_Bound$. Since there exists no solution to the sub-problem with value better than $Upper\_Bound$, there could be no solution to the original problem with value better than $Upper\_Bound$. Therefore the returned solution is optimal.

There are two main variants of the p-center problem in the literature; they differ by the possible location of the service points. Many authors deal with the *continuous* problem in which the points to be located optimally can be anywhere in the plane, but another interesting problem is the *discrete* case where there is a finite set of potential points (x_j, y_j) out of which one wishes to find the points which fulfill the minimax condition. In some cases, weights w_i are associated with the service points (a_i, b_i). Another classification of the problems is associated with the relevant metrics. In many cases, the distances between demand and service points are Euclidean (e.g. [9]). Also considered are problems where the distances are defined by minimal distances on a graph; this variant was first solved by Minieka [20].

The formulation of the Euclidean unweighted p-center problem is:

$$\min_{X_1,\dots,X_p} \left\{ \max_{1\le i\le n} \left[\min_{1\le j\le p} r_{ij} \right] \right\}$$

where $X_j = (x_j, y_j)$ for $j = 1, \dots, p$ is the location of the new facility and $r_{ij} = \left[(a_i - x_j)^2 + (b_i - y_j)^2\right]^{\frac{1}{2}}$. Megiddo and Supowit [19] have shown that both the p-center and p-median problems are NP-hard and that it is NP-hard even to approximate the p-center problems sufficiently closely. On the other hand, Hochbaum [14] has shown that given certain assumptions on the input distribution, there are polynomial algorithms that deliver a solution asymptotically close to the optimum with probability that is asymptotically one.

Most of the methods developed for solving the continuous Euclidean problem are geometrical in nature. When we are looking for a single service point ($p = 1$), the solution of the problem will be the center of the smallest circle enclosing n given points in the plane (see e.g. [6]). This can occur in one of two ways. The smallest circle can be determined by three demand points on its circumference or, alternatively, by two points on the two ends of a diameter. In the former case, the three points are the edges of an acute triangle [22]. The geometrical methods are based on a sophisticated search for the smallest enclosing circle among the circles built on subsets of two and three demand points. This includes the repeated solution of relaxed, smaller sub-problems as described below in the broader context of the p-center problem.

Chen [4] suggested a method that enables both the solution of the minisum and minimax location-allocation problems by using a differentiable approximation to the objective function and solving it by using nonlinear programming. This enabled the solution of relatively large problems, but the result was not necessarily optimal since local minima may have been reached. Drezner [9, 10] presented heuristic and optimal algorithms for the p-center problem in the plane. The heuristic method yielded results for problems with up to $n = 2000$ and $p = 10$ whereas the optimal method solved problems with up to $n = 30$, $p = 5$ or $n = 40$, $p = 4$. Watson-Gandy [27] suggested an algorithm that can optimally solve problems with up to about 50 demand points and 3 centers in reasonable time. The p-center problem on networks has been solved by Minieka [20] and by Toregas et al. [25]. A finite method, which is rather inefficient for large problems was suggested. An improvement based on the use of relaxations was offered by Handler and Mirchandani [13]. In section 2 we elaborate on relaxation methods.

Some other papers dealing with the continuous p-center problem include the following. Hwang et al. [15] describe a slab-dividing approach, which is expected to efficiently solve the Euclidean p-center problem. These authors show that their algorithm has time complexity of $O\left(n^{O(\sqrt{p})}\right)$. Suzuki and Drezner [24] propose heuristic procedures and upper bounds on the optimal solution where the demand points are distributed on a square. One of the methods they use employs the Voronoi heuristic. The same method has been recently used by Wei et al. [28]; the authors explore the complexity of solving the continuous space p-center problem in location planning. Agarwal and Sharir [1] discuss efficient approximate algorithms for geometric optimization, which includes the Euclidean p-center in d dimensions. Hale and Moberg [12] give a broad review on location problems, which includes the Euclidean p-center problem.

Another version of the p-center problem [11, 21] deals with the minimax p-center problem where the Euclidean distances are $d_{ij} = d(u_i, v_j)$ and where $U = \{u_1, u_2, \ldots, u_m\}$ is a set of m users and $V = \{v_1, v_2, \ldots, v_n\}$ a set of n potential locations for facilities in the plane. In these works, Tabu search and Variable Neighborhood Search methods as well as an optimal method are used, and the efficiency of these methods for small and large problems is evaluated. It should be noted that this Euclidean problem is equivalent to the p-center problem on networks where the possible location of the facilities are on the vertices and where the minimum distances between the demand and potential supply points are given. This discrete problem is also known to be NP-hard [18]. For a review on discrete network location models see Current et al. [7]. Recent works on these two versions of the discrete problem include algorithms given by Caruso et al. [3] and by Ilhan et al. [16]. The latter authors describe an efficient exact method for this p-center problem. Their algorithm finds the solution by updating, at each step, an upper or lower bound on the optimal solution. A tight lower bound to the optimal value is found in an initial phase of the algorithm, which consists of solving linear programming sub-problems. Good computational results are reported for each of an extensive list of test problems derived from OR-Lib and TSP-Lib problems with up to 900 data points.

In the present work, we present a new variant of existing relaxation algorithms for the continuous and discrete p-center problem. Past relaxation algorithms solved a set-covering problem at each step. We, on the other hand, solve a slightly easier problem at each step, namely Ilhan et al.'s feasibility sub-problem [16]. We have implemented this new variant of the relaxation algorithm[3]. Our experimental study shows that combining the strength of the relaxation method with the strength of state-of-the-art integer programming methods (used to solve the feasibility sub-problem) yields extremely efficient algorithms. For the discrete case, our algorithm often outperforms other optimal algorithms; for the continuous case, our algorithm solves problems which were previously considered too large. We analyze the advantages and disadvantages of relaxation methods compared to other optimal algorithms. We show that relaxation algorithms are best suited for problems where the number of demand points (n) is large, while the number of service points (p) is relatively small (for instance, $p < 20$).

We report computational results for both the continuous and discrete cases. In the continuous case, we show that for $n = 300$ demand points we can solve problems for any value of p ($1 \leq p \leq n$). Also, a "large" problem with $n = 1000$ and $p = 10$ is solved to optimality. In the discrete case, we solved problems taken from OR-Lib [2] and TSP-Lib [23] with up to 1817 demand points, and the run-times were compared to those reached by previous methods.

The rest of the paper is structured as follows. Section 2 explains the principles of relaxation and presents our own variant. In section 3 we present the results of our experimental study. Section 4 contains conclusions and open problems.

[3]Our code is publicly available at http://www.tau.ac.il/~chenr/or_research.html

2. Relaxation Algorithms for the P-Center Problem

2.1. **Theory.** Relaxation is a simple method to *optimally* solve a large location problem by solving a succession of small sub-problems. Although one cannot know in advance how many sub-problems need to be solved, once the global optimum is reached, it is identified as such. This as opposed to some heuristic methods which usually yield local minima. Though in the worst case, relaxation may be very slow, it is usually very efficient.

Chen and Handler [5] adapted the relaxation method to the problem in continuous Euclidean two-dimensional space. In the solution of the p-center problem there is usually only one circle which is critical in the sense that two or three demand points are on its circumference. There is much freedom in the exact position of the other circles and therefore, in the location of all but one of the centers. The value of the solution is determined by the radius of this critical circle, whereas the radii of the other circles may vary in size below this critical value. Thus, the number of possible optimal solutions is usually infinite. Chen and Handler [5] proved a theorem stating that among all the optimal solutions to the minimax problem of serving n demand points in Euclidean space by p service points, there is at least one in which all demand points are covered by critical circles, the largest of which has a radius r_p, which is the value of the solution. With the aid of this theorem, the search can be reduced to a finite number of critical circles.

The number of critical circles to be considered is $\binom{n}{3}+\binom{n}{2}+n$, where $\binom{n}{3}$ is the number of circles determined by three points on their circumference, $\binom{n}{2}$ is the number of circles defined by two points determining the diameter and n is the number of null circles; a null circle is a service point located at a demand point, the former serving only the latter. The number of possible combinations to cover n points by p critical circles becomes very large when n is large. However, geometrical considerations associated with a known upper bound and with the properties of relevant triangles defined by demand points, significantly reduce the size of the sub-problem to be solved.

The discrete case is slightly simpler. Each critical circle is defined by a single potential service point and a single demand point. Here too, we can significantly reduce the size of the sub-problem. We only consider a single circle for each potential service point: a circle centered at the service point and whose radius is the distance to the furthest demand point still within the upper bound.

2.2. **New Variant.** We now explain the changes we made to the relaxation algorithms [13, 5] so as to improve their performance.

The main change that we have made was to utilize Ilhan et al.'s feasibility sub-problem. Recall (from Algorithm 1) that at each step we need to solve:

$$Feasible \leftarrow \textsc{FindFeasibleSolution}(Sub, Upper\_Bound).$$

In other words, given a subset of the n demand points, we need to find a feasible (not necessarily optimal) solution to the relaxed (smaller) p-center problem, with value less than $Upper\_Bound$. As we have explained, in both the discrete and the continuous cases, we need only consider a finite number of critical circles in order to solve the problem. Therefore, our problem is reduced to checking whether, given a finite set of circles, there exist p circles that cover all of the demand points (in our relaxed problem).

How is this sub-problem solved? For each circle c we attach a binary vector v_c. The size of each vector is the number of demand points in the relaxed problem. For each circle c, the value of v_c in the position corresponding to demand point i

is 1 if and only if point i is covered by circle c:

$$v_c(i) = \begin{cases} 1 & \text{circle } c \text{ covers point } i \\ 0 & \text{otherwise} \end{cases} .$$

Finding p circles that cover all demand points is equivalent to finding p vectors such that their sum is greater than zero in all positions. This is almost exactly the set-covering problem, which can be stated as "find the minimal number of vectors whose sum is greater than the 1-vector". Minieka [20] was the first to propose using set-covering algorithms to solve the p-center problem. Subroutine FINDFEASIBLESOLUTION can be implemented by using a set-covering algorithm to find a minimal set of covering vectors, and checking whether the size of this set is less or equal to p (to determine whether a feasible solution exists).

Note that although set-covering algorithms return an optimal solution in terms of the minimal number of vectors, they do not return an optimal solution to the relaxed p-center problem. This is due to the fact that when we convert our circle-covering problem to the set-covering problem, we ignore the radius of each circle; we only take into account which demand points are within each circle. Therefore such implementation of FINDFEASIBLESOLUTION returns a feasible, not necessarily optimal, solution to the relaxed p-center problem.

Since we need not solve the relaxed p-center problem to optimality, we can utilize Ilhan et al.'s feasibility sub-problem [16] and avoid solving the set-covering problem to optimality. Rather than solving "find the minimal number of vectors whose sum is greater than the 1-vector", Ilhan et al. suggest solving a slightly easier problem, "find a subset of at most p vectors whose sum is greater than the 1-vector (if such a subset exists)". Although this problem is also NP-hard, eliminating the need to find a minimal set of vectors at each stage often improves the performance of the algorithm.

Chen and Handler [5] proposed to solve the p-center problem by first solving, optimally, the 1-center, then the 2-center, and eventually the $(p-1)$-center problem. The motivation is to start solving the p-center problem when you already have a reasonably tight upper bound on the solution. It turns out that when using state-of-the-art algorithms for the set-covering problem (or in our case, the feasibility sub-problem), this approach hinders, rather than improves, performance. We have improved performance by starting-off, immediately, solving the p-center problem.

3. EXPERIMENTAL RESULTS

3.1. **Methodology.** For the discrete p-center problem, we compare the performance of our relaxation code to that of Ilhan et al.'s [16], which is, to the best of our knowledge, the best code currently available for the optimal solution of the discrete p-center problem. Ilhan et al. use an iterative algorithm which is a variant of Minieka's algorithm [20].

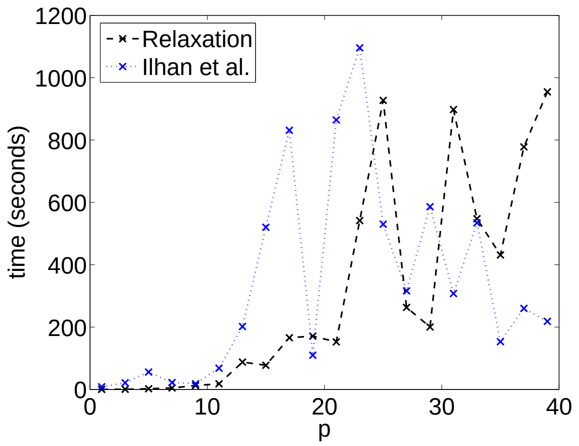
Comparing our algorithm to that of Ilhan et al. gives us insight to the strengths and weaknesses of relaxation, although it should be noted that we made little effort to optimize either code.

3.2. **Experimental Setup.** The experiments were conducted on a 3.2 GHz Pentium 4 computer with 2 GB of main memory. The computer runs the Linux 2.6.17. The code is written in C and compiled using gcc-4.0.

The feasibility sub-problem can be formulated as an Integer Programming problem[4]. We used CPLEX version 7.5 [17] for the solution of the Integer Programming problems. CPLEX implements optimizers based on the simplex algorithms. We

[4]This is also true for the p-center problem and the set-covering problem.

FIGURE 1. The performance of Ilhan et al.'s algorithm and our relaxation algorithm on a discrete problem with Euclidean distances (d657.tsp) with $n = 657$ demand points.



found that best results are achieved when CPLEX uses the primal Simplex algorithm for our Integer Programming sub-problems.

3.3. **Experimental Analysis.** Relaxation-based iterative algorithms represent a different approach to previous optimal iterative algorithms for the discrete p-center problem, such as those suggested by Ilhan et al. [16], Daskin [8] and Elloumi et al. [11]. The traditional iterative algorithms update, at each step, an upper or lower bound on the optimal solution, until the optimal solution is reached.

Roughly speaking, while traditional algorithms need to solve few large problems, relaxation algorithms need to solve many small problems.

Our experiments show that relaxation is particularly suited for problems with relatively small values of p (for instance, $p < 20$). Figure 1 shows the results for a typical problem; relaxation performs very well for smaller values of p, but as p grows larger, it loses its advantage over traditional iterative algorithms. The graph shows the performance of Ilhan et al.'s algorithm and our relaxation algorithm on problem d657.tsp, taken from TSP-Lib [23], for $p = 1, 3, 5, \ldots, 39$. The figure also demonstrates that the behavior of both algorithms is rather unpredictable.

For the continuous p-center problem, relaxation enables us to solve problems which were previously considered too large.

3.3.1. *Discrete p-center.* Like Ilhan et al. [16], we tested the performance of our relaxation algorithm on the p-median (pmed) inputs[5] from OR-Lib [2]. We have found that the relaxation algorithm solved the entire 40 problems in 55.02 seconds, whereas Ilhan et al.'s algorithm (the Minieka sub-problem variant) took 83.58 seconds. We get a better understanding of the advantages and disadvantages of relaxation algorithms when we sort the problems by the values of p. We divided the pmed problems to three: problems with "small" p values ($p = 5, 10$), "medium" p values ($20 \leq p \leq 67$), and "large" p values ($70 \leq p \leq 200$). Table 1 sums up the total running times for the "small", "medium", and "large" sets of problems. For complete information on the performance of relaxation versus Ilhan et al.'s algorithm on the pmed problems, see the appendix.

[5]In these problems, the set of demand points and the set of potential service points are the same.

TABLE 1. Total running times of Ilhan et al.'s algorithm and our relaxation algorithm on discrete pmed problems.

	Ilhan et al. (secs)	relaxation (secs)
small	58.73	7.16
medium	5.44	9.46
large	19.41	38.4

TABLE 2. The performance of our relaxation algorithm on large discrete problems from TSP-Lib. The two rightmost columns are for the maximal size of a relaxation sub-problem, and the total number of relaxation sub-problems.

input	n	p	obj	relaxation (secs)	max prob	no. sub
rl1323	1323	5	4543	7.82	119	214
rl1323	1323	10	3077	263.13	390	613
rl1323	1323	15	2347	822.49	593	967
rl1323	1323	20	2016	4866.53	762	1268
rl1323	1323	25	1765	52939.65	836	1452
u1817	1817	5	715	6.4	117	200
u1817	1817	10	458	885.68	475	653
u1817	1817	15	359	50218.35	665	888
u1817	1817	20	309	153036.37	870	1119

TABLE 3. Total running times of Ilhan et al.'s algorithm and our relaxation algorithm on discrete problems with Euclidean distances from TSP-Lib.

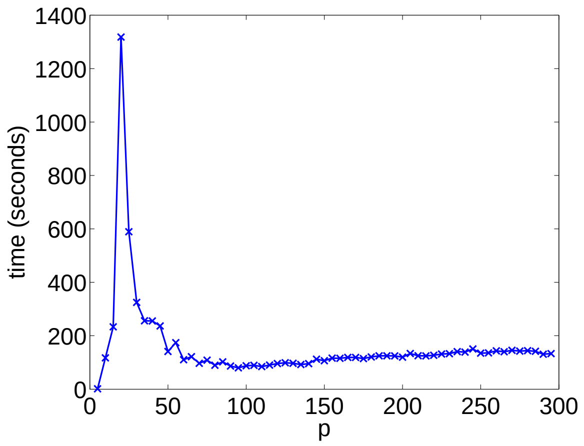
	Ilhan et al. (secs)	relaxation (secs)
$p = 5$	89.82	8.17
$p = 10$	64.92	48.73
$p = 20$	120.77	248.9
$p = 40$	457.61	808.61

Like Elloumi et al. [11], we tested the performance of our relaxation algorithm on problems rl1323 and u1817 from TSP-Lib [23]. Given the coordinates of the demand points in the plane, the Euclidean distance is computed and rounded to the nearest integer for every pair of demand points. Besides running times, table 2 shows, for each problem, the size of the maximal sub-problem solved by the relaxation algorithm, and the total number of sub-problems solved by the relaxation algorithm.

3.3.2. *Discrete p-center problems with Euclidean distances.* Like Ilhan et al. [16], we tested the performance of our relaxation algorithm on problems from TSP-Lib [23]. Table 3 sums up the total running times for $p = 5, 10, 20, 40$. For more details on the performance of relaxation versus Ilhan et al.'s algorithm on the TSP-Lib problems, see the appendix.

3.3.3. *Continuous p-center problems with Euclidean distances.* Combining the power of relaxation with a state-of-the-art integer programming software (CPLEX), allows

FIGURE 2. Running times of relaxation on a random continuous p-center problem with $n = 300$.



us to solve continuous p-center problems which were previously considered too large.

Figure 2 shows the running times of relaxation on a problem with $n = 300$ demand points randomly chosen in a square. We solve the problem for $p = 5, 10, \ldots, 295$ service points. For complete information on the performance of relaxation on this problem, see the appendix.

As graph 2 shows, when p is sufficiently large, the p-center problem becomes relatively easy. That is due to the fact that the bounds obtained from feasible candidate solutions are small enough to rule out most of the critical circles. It is interesting to note that for $p \geq 16$, the size of the maximal relaxation sub-problem we need to solve is greater than 200, which is not particularly small compared to $n = 300$. Relaxation works well in these cases because the earlier stages, when the sub-problems are small, yield tight bounds on the solution, which simplify the problems we need to solve as the relaxation subsets become bigger.

We have run our algorithm on a random continuous p-center problem with $n = 1000$ and $p = 10$. After about 27 hours, it has still not reached the solution. At that point the value of the best candidate solution was 20.292586. We ran the algorithm again with an initial upper bound of 20.292586; the algorithm solved the problem to optimality within twelve minutes (the optimal value was 20.001561). This suggests two things:

- Having a tight upper bound on the solution greatly improves the running time.
- In problems of this magnitude ($n = 1000$), once the relaxation sub-problems become too large, it is a good idea to abandon the current relaxation subset, and to start over with a small new random subset, using the value of the current candidate solution as an upper bound.

4. CONCLUSIONS AND OPEN PROBLEMS

4.1. **Conclusions.** Relaxation is a simple method to *optimally* solve a large location problem by solving a succession of small sub-problems.

We presented new variants of relaxation, for the discrete and continuous p-center problems. We have conducted an experimental study that demonstrated that these new variants are very efficient. For the discrete case, our algorithm often outperforms other optimal algorithms; for the continuous case, the algorithm solves problems which were previously considered too large.

Our experiments show that relaxation is particularly suited for problems with small values of p. When p is not very large (for instance, $p < 20$), we need to solve relatively few small sub-problems until the optimal solution is reached. As p grows larger, the number and the size of the sub-problems grow larger too, and relaxation may lose its advantage over traditional iterative algorithms.

4.2. **Open Problems.** One remaining question is whether our relaxation algorithms can be improved by applying a heuristic to efficiently find a feasible solution, and thus provide a tight initial upper bound on the solution.

Another open question is whether we can employ Vijay's method [26] to solve the reduced problems and improve our relaxation algorithm for the continuous case.

It would also be interesting to investigate heuristics which determine the points in time in which it is beneficial to abandon the current relaxation subset (which could get extremely large), and to start over with a small new random subset, using the value of the current feasible solution as an upper bound.

Acknowledgments. Thanks to Taylan Ilhan for sending us his code for the discrete p-center problem. Thanks to Gil Shklarski, Sivan Toledo and Leon Ankonina for their help with the utilization of CPLEX.

References

[1] P. K. Agarwal and M. Sharir. Efficient algorithms for geometric optimization. *ACM Comut. Surv.*, 30:428–458, 1998.
[2] J. E. Beasley. A note on solving large p-median problems. *Eur. J. Oper. Res.*, 21:270–273, 1985. http://people.brunel.ac.uk/~mastjjb/jeb/orlib/pmedinfo.html.
[3] C. Caruso, A. Colorni, and L. Aloi. Dominant, an algorithm for the p-center problem. *Eur. J. Oper. Res.*, 149:53–64, 2003.
[4] R. Chen. Solution of minisum and minimax location-allocation problems with Euclidean distances. *Naval Res. Log. Quart.*, 30:449–459, 1983.
[5] R. Chen and G. Y. Handler. Relaxation method for the solution of the minimax location-allocation problem in Euclidean space. *Nav. Res. Log.*, 34:775–787, 1987.
[6] G. Chrystal. On the problem to construct the minimum circle enclosing n given points in the plane. *Proc. Edinburgh Math. Soc.*, 3:30–33, 1885.
[7] J. Current, M. Daskin, and D. Schilling. Facility location: Applications and theory. In Z. Drezner and H. W. Hamacher, editors, *Discrete network location models*, pages 83–120. Springer, 2001.
[8] M. S. Daskin. *Network and discrete location, models: algorithms and applications*. Wiley Interscience Pub., John Wiley and Sons Inc., New-York, 1995.
[9] Z. Drezner. The p-center problem – heuristic and optimal algorithms. *J. Oper. Res.*, 8:741–748, 1984.
[10] Z. Drezner. The planar two center and two median problems. *Trans. Sci.*, 18:351–361, 1984.
[11] S. Elloumi, M. Labbé, and Y. Pochet. A new formulation and resolution method for the p-center problem. *INFORMS J. Comput.*, 16:84–94, 2004.
[12] S. H. Hale and C. R. Moberg. Location science reserch: A review. *Ann. Oper. Res.*, 123:21–35, 2004.
[13] G. Y. Handler and P. B. Mirchandani. *Location on networks: Theory and algorithms*. MIT Press, Cambridge, MA, 1979.
[14] D. S. Hochbaum. When are NP-hard problems easy? *Ann. Oper. Res.*, 1:201–214, 1984.
[15] R. Z. Hwang, R. C. T. Lee, and R. C. Cheng. The slab dividing approach to solve the Euclidean p-center problem. *Algorithmica*, 9:1–22, 1993.
[16] T. Ilhan, F. A. Özsoy, and M. C. Pinar. An efficient exact algorithm for the vertex p-center problem and computational experiments for different set covering subproblems. Technical report, 2002. Available at http://www.optimization-online.org/DB_HTML/2002/12/588.html.
[17] ILOG, Inc, Gentilly, France. *ILOG CPLEX 7.5 User's Manual*, November 2001.
[18] O. Kariv and S. L. Hakimi. An algorithmic approach to network location problems. part 1: p-centers. *SIAM J. Appl. Math.*, 37:513–538, 1971.
[19] N. Megiddo and K. J. Supowith. On the complexity of some common geometric location problems. *SIAM J. Comput.*, 13:182–196, 1984.
[20] E. Minieka. The m-center problem. *SIAM Rev.*, 12:139–140, 1970.
[21] N. Mladenović, M. Labbé, and P. Hansen. Solving the p-center problem with tabu search and variable neighborhood search. *Networks*, 42:48–64, 2003.
[22] H. Rademacher and O. Toeplitz. The spanning circle of a finite set of points. In *The enjoyment of mathematics*, pages 103–110. Princton University Press, Princeton, NJ, 1957.
[23] G. Reinelt. TSP-Lib. http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsplib.html.
[24] A. Suzuki and Z. Drezner. The p-center location problem in the area. *Location Sci.*, 4:69–82, 1996.

[25] C. Toregas, R. Swain, C. ReVelle, and L. Bergmann. The location of emergency service facilities. *Oper. Res.*, 19:1363–1373, 1971.
[26] J. Vijay. An algorithm for the p-center problem in the plane. *Trans. Sci.*, 19:235–245, 1985.
[27] C. D. T. Watson-Gandy. The multi-facility min-max Weber problem. *Eur. J. Oper. Res.*, 18:44–50, 1984.
[28] H. Wei, A. T. Murray, and N. Xiao. Solving the continuous space p-center problem: Planning applications issues. *IMA J. Manag. Math.*, 17:413–425, 2006.

TABLE 4. The performance of Ilhan et al.'s algorithm and our relaxation algorithm on discrete problems with "small" values of p. The two rightmost columns are for the maximal size of a relaxation sub-problem, and the total number of relaxation sub-problems.

input	n	p	obj	Ilhan et al. (secs)	relaxation (secs)	max prob	no. sub
pmed1	100	5	127	0.27	0.12	23	51
pmed6	200	5	84	0.38	0.19	23	52
pmed11	300	5	59	0.81	0.19	19	35
pmed16	400	5	47	1.15	0.19	17	33
pmed21	500	5	40	1.82	0.45	27	48
pmed26	600	5	38	4.35	0.23	21	36
pmed31	700	5	30	3.86	0.27	24	39
pmed35	800	5	30	4.88	0.17	14	25
pmed38	900	5	29	7.75	0.25	18	36
pmed2	100	10	98	0.11	0.27	38	96
pmed3	100	10	93	0.09	0.3	38	107
pmed7	200	10	64	0.25	0.24	30	66
pmed12	300	10	51	0.68	0.28	33	64
pmed17	400	10	39	1.18	0.5	45	72
pmed22	500	10	38	2.54	1.36	57	87
pmed27	600	10	32	3.42	0.34	34	52
pmed32	700	10	29	11.01	0.59	41	59
pmed36	800	10	27	5.33	0.78	46	61
pmed39	900	10	23	8.85	0.44	35	49

APPENDIX A. TABLES

A.1. **Discrete p-center problems.** Tables 4, 5 and 6 show the performance of relaxation versus Ilhan et al.'s algorithm for problems with "small", "medium", and "large" values of p respectively. The input files are taken from OR-Lib [2]. Besides running times, the tables show, for each problem, the size of the maximal sub-problem solved by the relaxation algorithm, and the total number of sub-problems solved by the relaxation algorithm.

A.2. **Discrete p-center problems with Euclidean distances.** Table 7 shows the performance of relaxation versus Ilhan et al.'s algorithm on discrete p-center problems with Euclidean distances. The input files are taken from TSP-Lib [23].

A.3. **Continuous p-center problems with Euclidean distances.** Table 8 shows the performance of our relaxation algorithm on a random continuous p-center problem with $n = 300$ demand points.

TABLE 5. The performance of Ilhan et al.'s algorithm and our relaxation algorithm on discrete problems with "medium" values of p.

input	n	p	obj	Ilhan et al. (secs)	relaxation (secs)	max prob	no. sub
pmed4	100	20	74	0.08	0.32	48	142
pmed8	200	20	55	0.18	0.37	58	121
pmed13	300	30	36	0.4	0.7	77	124
pmed5	100	33	48	0.08	0.37	50	147
pmed9	200	40	37	0.17	0.57	84	156
pmed18	400	40	28	0.74	1.21	118	158
pmed23	500	50	22	0.89	2.4	153	194
pmed14	300	60	26	0.3	1.11	136	203
pmed28	600	60	18	2.49	1.77	163	192
pmed10	200	67	20	0.11	0.64	97	170

TABLE 6. The performance of Ilhan et al.'s algorithm and our relaxation algorithm on discrete problems with "large" values of p.

input	n	p	obj	Ilhan et al. (secs)	relaxation (secs)	max prob	no. sub
pmed33	700	70	15	3.92	3.42	222	250
pmed19	400	80	18	0.38	2.06	188	233
pmed37	800	80	15	3.49	3.81	245	276
pmed40	900	90	13	7.11	7.2	314	344
pmed15	300	100	18	0.23	1.16	155	223
pmed24	500	100	15	0.61	2.77	206	253
pmed29	600	120	13	0.8	3.01	264	300
pmed20	400	133	13	0.35	2.68	244	301
pmed34	700	140	11	1.24	3.76	284	327
pmed25	500	167	11	0.55	3.52	262	312
pmed30	600	200	9	0.73	5.01	357	409

TABLE 7. The performance of Ilhan et al.'s algorithm and our relaxation algorithm on discrete p-center problems with Euclidean distances.

input	n	p	obj	Ilhan et al. (secs)	relaxation (secs)	max prob	no. sub
kroA200.tsp	200	5	911.412091	0.29	0.36	43	89
kroB200.tsp	200	5	897.669204	0.39	0.24	30	80
gr202.tsp	202	5	19.384514	12.23	0.1	11	38
pr226.tsp	226	5	3720.551034	0.7	0.41	27	102
pr264.tsp	264	5	1610.124219	0.42	0.16	15	46
pr299.tsp	299	5	1336.272801	0.71	0.95	58	160
lin318.tsp	318	5	1101.339639	0.66	0.58	37	94
pr439.tsp	439	5	3196.580204	1.71	0.71	35	85
pcb442.tsp	442	5	1024.74387	2.32	1.79	87	159
d493.tsp	493	5	752.908474	14.75	0.77	38	80
d657.tsp	657	5	880.908537	55.64	2.1	70	175
kroA200.tsp	200	10	598.819672	0.37	1.19	92	209
kroB200.tsp	200	10	582.103943	0.35	1.69	111	256
gr202.tsp	202	10	9.334002	10.67	0.39	38	105
pr226.tsp	226	10	2326.478025	0.21	0.5	34	141
pr264.tsp	264	10	850	0.31	0.46	35	104
pr299.tsp	299	10	888.835755	0.84	2.05	114	252
lin318.tsp	318	10	743.210603	0.52	2.32	106	244
pr439.tsp	439	10	1971.832904	1.39	1.49	64	160
pcb442.tsp	442	10	670.820393	3.57	9.5	194	302
d493.tsp	493	10	458.304571	13.16	2.97	95	228
d657.tsp	657	10	574.744682	33.53	26.17	223	404
kroA200.tsp	200	20	389.307077	0.21	1.9	129	367
kroB200.tsp	200	20	382.280002	0.79	2.23	139	415
gr202.tsp	202	20	5.56569	9.08	1.29	99	286
pr226.tsp	226	20	1365.650028	0.27	0.68	49	185
pr264.tsp	264	20	514.781507	0.28	0.87	80	175
pr299.tsp	299	20	559.016994	0.43	3.58	171	429
lin318.tsp	318	20	496.452415	0.4	2.65	129	352
pr439.tsp	439	20	1185.59057	0.99	2.65	107	286
pcb442.tsp	442	20	447.213595	1.73	16.65	286	471
d493.tsp	493	20	312.744896	55.73	12.47	199	454
d657.tsp	657	20	374.7	50.86	203.93	446	894
kroA200.tsp	200	40	258.259559	0.2	2.68	150	555
kroB200.tsp	200	40	253.237043	0.19	2.89	157	584
gr202.tsp	202	40	2.971363	7.01	2.93	157	564
pr226.tsp	226	40	650	0.21	1.13	91	275
pr264.tsp	264	40	316.227766	20.64	443.51	172	284
pr299.tsp	299	40	355.31676	0.35	4.2	215	567
lin318.tsp	318	40	315.919293	0.66	5.66	188	531
pr439.tsp	439	40	671.751442	0.99	6.92	227	575
pcb442.tsp	442	40	316.227766	140.48	13.1	318	560
d493.tsp	493	40	206.015655	2.03	17.94	321	781
d657.tsp	657	40	249.51541	284.85	307.65	530	1320

TABLE 8. The performance of our relaxation algorithm on a random continuous p-center problem with $n = 300$ demand points.

p	obj	relaxation (secs)	max prob	no. sub
5	28.422647	1.37	54	88
10	18.772435	117.35	143	220
15	14.465007	233.42	193	301
20	12.192055	1318.11	236	393
25	10.48132	589.28	252	414
30	9.251168	324.95	264	429
35	8.402865	256.11	266	437
40	7.683381	255.31	271	464
45	7.135402	236.86	275	497
50	6.429138	141.35	282	496
55	6.027485	174.23	281	529
60	5.622641	110.45	284	514
65	5.241039	121.69	286	539
70	4.943647	96.94	286	538
75	4.701547	108.66	290	575
80	4.521751	89.6	285	551
85	4.226306	102.29	288	603
90	4.087438	86.77	289	582
95	3.789379	79.97	287	568
100	3.593477	87.73	285	608
105	3.505622	89.29	296	592
110	3.395071	84.73	292	588
115	3.305871	90.23	293	658
120	3.159556	96.07	293	691
125	3.049413	98.83	294	684
130	2.936309	97.13	296	679
135	2.819311	92.42	294	665
140	2.745859	95.8	294	679
145	2.575867	112.43	297	745
150	2.464594	106.62	297	721
155	2.39577	116.26	297	764
160	2.34536	115.73	297	762
165	2.304482	118.92	297	819
170	2.195573	118.99	297	814
175	2.133524	114.46	296	818
180	2.024234	120.55	297	829
185	1.932473	125.02	297	851
190	1.834879	124.79	297	847
195	1.759475	124.44	297	846
200	1.639179	119.95	298	837
205	1.598125	133.63	299	885
210	1.53924	125.11	298	869
215	1.472276	124.68	297	874
220	1.436264	127.07	297	882
225	1.406136	131.73	299	890
230	1.336306	132.94	299	893
235	1.281722	140.46	299	926
240	1.228482	138.9	299	937

p	obj	relaxation (secs)	max prob	no. sub
245	1.135185	150.61	299	977
250	1.095122	134.79	298	945
255	0.993977	136	298	949
260	0.932198	143.69	300	960
265	0.864703	140.1	299	946
270	0.804964	145.64	300	969
275	0.718482	142.42	300	953
280	0.685738	144.15	299	987
285	0.525316	141.82	300	1000
290	0.405084	130.82	300	964
295	0.310794	133.4	300	985