# Incorporating Neighborhood Reduction for the Solution of the Planar $p$-Median Problem

Zvi Drezner
Steven G. Mihaylo College of Business and Economics
California State University-Fullerton
Fullerton, CA 92834.
e-mail: zdrezner@fullerton.edu

Said Salhi
Centre for Logistics & Heuristic Optimization
Kent Business School
University of Kent, Canterbury CT2 7PE, United Kingdom.
e-mail: S.Salhi@kent.ac.uk

**Abstract**

Two efficient neighborhood reduction schemes are proposed for the solution of the $p$-Median problem on the plane. Their integration into a local search significantly reduces the run time with an insignificant deterioration in the quality of the solution. For completeness this fast local search is also embedded into one of the most powerful metaheuristic algorithms recently developed for this continuous location problem. Excellent results for instances with up to 1060 demand points with various values of $p$ are reported. Eight new best known solutions for ten instances of a large problem with 3,038 demand points and up to 500 facilities are also found.

**Keywords-** planar $p$-median, local search, neighborhood reduction, metaheuristics

## 1 Introduction

The location allocation problem in the plane, sometimes called the continuous p-median or the multi-source Weber problem is to find $p$ locations for facilities in the plane to provide service to a set $N$ of $n$ demand points each with an associated weight $w_i > 0$. Each demand point gets its service from the closest facility to it. The objective is to minimize the total sum of weighted minimum distances to the facilities. Let $d_i(X_j)$ be the Euclidean distance between demand point $i$ and facility $j$ located at $X_j = (x_j, y_j)$. The vector of unknown locations is $X = \{X_1, \ldots, X_p\}$, and thus, the objective function to be minimized is:

$$F(X) = \sum_{i \in N} w_i \min_{1 \le j \le p} \{d_i(X_j)\} \tag{1}$$

Drezner [12] and Chen et al. [8] developed optimal solution procedures for the location of 2 facilities whereas Schöbel and Scholz [29] optimally solved problems with 3 facilities. Krau [21] used column generation to optimally solve problems with $n = 50$ and 287 demand points and any number of facilities.

The continuous (also known as planar) $p$-median problem with the objective function (1) is known to be NP-hard [22], and as a result, many heuristics have been developed for its solution. Classical heuristics include the famous alternating procedure by Cooper [10, 11], the projection method of Bongartz et al. [2], and gradient-based methods such as Chen [9], Murtagh and Niwattisyawong [25]. The leading heuristics to date for solving this problem are based on variable neighborhood search (e.g. see [7, 14]). For larger scale problem instances, decomposition strategies are successfully applied [5, 30]. Further gains may be obtained by using new local searches (e.g., [13]) or variable neighborhood descent within a general variable neighborhood search [23]. The best results to date were obtained by [14]. For recent reviews of solution approaches to the continuous $p$-median problem the reader is referred to [4, 6, 13, 14].

The contribution of the paper is the design and analysis of efficient neighborhood reduction schemes as part of an already powerful local search for solving the planar $p$-median problem. We shall demonstrate the impact of such reduction schemes on several data sets while identifying several new best results for this important and well researched continuous location problem.

The paper is organized as follows. The next section describes the design of the two neighborhood reduction schemes which are embeded within a well known local search for the continuous $p$-median. This is followed by a section on computational results that comprises sensitivity analysis, intensive experiments showing the effect of these reduction tests when embedded into a recently developed powerful metaheuristic. The final section summarizes our findings.

## 2 Applying a neighborhood reduction scheme to the IMP algorithm

We first briefly describe the IMP algorithm that will incorporate the reduction tests that we propose. For more details on this algorithm and other powerful metaheuristics for the $p$-median problem on the plane, see [13, 14].

*The IMP algorithm-* Suppose that starting locations of the $p$ facilities are given. The optimal location for a facility $1 \leq k \leq p$ while holding the other $p-1$ facilities rooted in their locations can be found as follows. The shortest distance between demand point $i$ and the facilities is the minimum between the unknown distance to facility $k$ and the minimum distance to all other fixed facilities which is fixed and does not depend on the location of facility $k$. Define the minimum distance of $i \in N$ to the fixed facilities as

$$D_i = \min_{j \neq k} \{d_i(X_j)\}. \tag{2}$$

The idea is to find the best location for facility $k$, $X_k$, while holding all other facilities fixed by minimizing $G(X)$:

$$G(X) = \sum_{i \in N} w_i \min \{d_i(X), D_i\}. \tag{3}$$

where $D_i$ are constants defined by (2).

Problem (3) is the limited distance location problem (LD) that can be formulated for any number of facilities. In this study, we applied the single facility version of the problem which was introduced in [15]. This problem can be optimally solved by global optimization techniques such as "Big Square Small Square" (BSSS) proposed by Hansen et al. [20] and improved by Plastria [26]. Other global optimization algorithms are "Big Triangle Small Triangle" (BTST) by Drezner and Suzuki [16] for planar problems, "Big Cube Small Cube" (BCSC) of Schöbel and Scholz [29] for location problems in higher dimensions, and "Big Segment Small Segment" by Berman et al. [1] for locating a facility anywhere on the network. The complete solution procedure for the IMP algorithm is given in [13] where instances with up to 100,000 demand points are optimally solved by BSSS in less than 3 seconds of computing time.

It was empirically shown that the IMP algorithm provides much better results than the Cooper-style alternate algorithm ALT or its improved version IALT [3, 10, 11]. In brief, in both ALT and IALT the value of $D_i$ can be considered to be $\infty$ which further lowers the quality of the solution.

The LD problem is formulated by using all $n$ demand points. Our proposed neighborhood reduction scheme attempts to reduce the number of demand points in the LD formulation thus reducing the required computing time. There are two cases concerning the selection of demand points to be included in the LD formulation for relocating the open facility $k$. Let $S$ be the set of all the demand points presently served by facility $k$.

Case 1: When the optimal location of facility $k$ is changed, but remains in the convex hull of $S$, demand points near the periphery of $S$ may become closer to a fixed facility and thus will not be served by facility $k$, while points not in $S$ may become closer to facility $k$ and thus will be served by it.

Case 2: When the removed facility does not serve much demand (i.e., its removal does not affect the value of the objective function by much), it is likely that the optimal location is in a different region that may be far from $S$. This is because when locating facility $k$ at the optimal location, the value of the objective function may decrease more than the increase following its removal.

The aim is to produce a rule that distinguishes between these two cases. For instance when Case 1 happens, the number of demand points used in the LD algorithm would be reduced significantly by eliminating those demand points that are not likely to be included in the LD solution, i.e., $d_i(X^*) > D_i$ for the optimal location $X^*$ of (3). To achieve this, we propose reduction schemes for Case 1, also known as neighboring reduction rules. The idea is to minimize the risk of excluding good quality solutions while eliminating those inferior alternatives that are not worth considering. To achieve this goal, we need to examine these two cases concerning the optimal location $X^*$ of the LD problem.

## 2.1 Case 1: $X^*$ is in the convex hull of $S$.

In this case, most of the demand points in $S$ will be serviced by facility $k$ relocated to $X^*$. The set of demand points included in the LD problem should include (i) the set $S$, even though some points in $S$ may not be part of the LD solution, and (ii) some additional demand points which are not in $S$ but close to the

periphery of the set $S$ as these may become closer to $X^*$ than to their original nearest fixed facilities. The following two neighborhood reduction rules are proposed for identifying such demand points.

**Rule 1: The Convex Hull (CH) Rule**

Let the largest distance from $X_k$ to all demand points in $S$ be $d_{\max}$.

$$d_{\max} = \max_{i \in S} \{d_i(X_k)\} \tag{4}$$

Consider a demand point $i$ not in $S$. Its shortest distance to a fixed facility is $D_i$, and its distance to facility $k$ is $d_i(X_k) > D_i$. Demand point $i$ will not be closest to $X^*$ if its nearest point in the convex hull of $S$ is farther than $D_i$. The shortest distance to the convex hull is estimated as the shortest distance to its vertices. By the triangle inequality, the distance to $X^*$ is at least $d_i(X_k) - d_{\max}$. All demand points that satisfy $d_i(X_k) - d_{\max} > D_i$ can be eliminated from the set of demand points and hence the LD problem can be based on the set

$$\overline{S} = \{i \in N \mid d_i(X_k) \leq D_i + d_{\max}\}. \tag{5}$$

This definition of $\overline{S}$ can be generalized by introducing a parameter $\theta \geq 0$:

$$\overline{S}(\theta) = \{i \in N \mid d_i(X_k) \leq D_i + \theta d_{\max}\}. \tag{6}$$

Note that $\overline{S}(0) = S$, $\overline{S}(1) = \overline{S}$, and for a large $\theta$, $\overline{S}(\theta) = N$. In general, for $\theta_1 \leq \theta_2$, $\overline{S}(\theta_1) \subseteq \overline{S}(\theta_2)$. For instance, in ALT [10, 11] or IALT [3], the set $\overline{S}(\theta) = S$ (i.e., $\theta = 0$).

**Rule 2: The Near Boundary (NB) Rule**

Consider the ratio $r = \frac{d_i(X_k)}{D_i}$ for demand point $i$. When $r < 1$ demand point $i$ is in $S$. When $r = 1$ demand point $i$ is equidistant to facility $k$ and another facility and can be served by both. When $r > 1$ demand point $i$ is not in $S$. However, when $r$ is close to 1 demand point $i$ is close to the bisector separating facility $k$ from another "close" facility. These points are also known in the multi-depot vehicle routing literature as the borderline customers as these can be assigned to either their nearest or their second or third nearest depot (Golden et al. [18] and Salhi and Sari [28]). In our case, if facility $k$ is relocated in the general direction of demand point $i$, this customer may be serviced by facility $k$ rather than its present server. We would like to extend the set $S$ to include points that are close to the boundary of $S$. A parameter $\rho > 1$ is chosen and the set $\overline{\overline{S}}(\rho)$ is defined as

$$\overline{\overline{S}}(\rho) = \{i \in N \mid d_i(X_k) \leq \rho D_i\}. \tag{7}$$

Note that if $\rho = 1, \overline{\overline{S}}(1)$ reduces to $\overline{S}(0) = S$. Also, Equation (7) is similar to (6) and the two expressions become identical when $\rho = 1 + \theta \frac{d_{\max}}{D_i}$. However, this entails using different $\rho$'s for different demand points.

## 2.2 Case 2: $X^*$ is in another region of the solution space.

In this case all $n$ demand points should be included in the LD formulation and no run time is saved. Actually, if the solution is in the convex hull of $S$, the optimal solution will also be found when all $n$ demand points are

included. Therefore, including all points in $N$ will not affect the optimal solution but a potential reduction in run time could be lost.

## 2.3 A Rule for Distinguishing Between the Two Cases

A rule has to be crafted so that the correct situation (whether $X^*$ is in the convex hull or not) is identified in the vast majority of cases. We view the process as two consecutive steps. First, facility $k$ is removed and then a facility at $X^*$ is added to the $p-1$ fixed facilities. The increase in the value of the objective function when facility $k$ is removed, $\Delta F$, is:

$$\Delta F = \sum_{i \in S} w_i \left[ D_i - d_i(X_k) \right] \tag{8}$$

The change in the value of the objective function is $\Delta F$ minus the reduction in the value of the objective function when a facility at $X^*$ is added. If $\Delta F$ is small, there is a good chance that $X^*$ is in a different region of the plane. On the other hand, if $\Delta F$ is relatively large, $X^*$ is likely to be close to $X_k$. The value of the objective function is unchanged when $X^*$ is at $X_k$. If this location is not a local optimum, a small move in its location should improve the value of the objective function. There must be a relatively large reduction in the value of the objective function at some $X^*$ in a different region to obtain a better solution.

Let $F$ be the value of the objective function. The expected contribution of a facility to $F$ is $\frac{F}{p}$. We introduce a correction factor $\alpha$ ($\alpha \geq 0$) and set the threshold of $\alpha \frac{F}{p}$ to determine whether to apply case 1 or case 2 when solving the LD problem. This implementation is summarized in the following algorithm.

**The Fast LD Algorithm**

The parameters $\alpha \geq 0$ and $\theta \geq 0$ (or $\rho = 1 + \theta \geq 1$) are given.

1. Calculate $\Delta F$ by (8), evaluate the value of the objective function $F$ by (1) and set $\overline{S} = N$.
2. If $\Delta F \geq \alpha \frac{F}{p}$, then
   (a) By the CH rule evaluate $\overline{S} = \overline{S}(\theta)$ using (6).
   (b) By the NB rule evaluate $\overline{S} = \overline{\overline{S}}(\rho)$ using (7).
3. Solve LD including all demand points in $\overline{S}$.

When $\alpha$ is large, all demand points will be included in the LD problem, and there will be no time saving. When $\alpha$ is too small, the quality of the LD solution may not be satisfactory as some combinations may not be considered for re-assignment and hence limits the search. In the next section, we identify empirically a value of $\alpha$ that will save computational time while maintaining or having a small deterioration in the quality of the solutions.

In summary, the IMP procedure which applies the fast LD procedure given above for solving the LD problem is referred to as the fast IMP (FIMP for short). Also note that the initial square when applying IMP for the BSSS algorithm is the smallest square enclosing all demand points, but when the set $\overline{S}$ is applied in FIMP, the smallest square enclosing $\overline{S}$ can be used as the initial square.

# 3 Computational Experiments

The programs were coded in Fortran using double precision arithmetic and compiled by an Intel 11.1 Fortran Compiler with no parallel processing. They were run on a desktop with the Intel 870/i7 2.93GHz CPU Quad processor and 8GB RAM using only one thread.

We tested the above ideas on three problems with $n = 654, 1060$ and 3038 from the TSP library (Reinelt [27]). The first two problems were tested in [7] and the last one in [5, 30]. Note that two smaller problems with $n = 50$ from [17] and $n = 287$ from [2], for which we know the optimal solutions [21], were also tested in previous papers. However, the starting solutions of 100 random generations are found to be optimal in all experiments and hence there is no need to test these two smaller problems.

## 3.1 A Sensitivity analysis

We investigated the sensitivity of the fast LD procedure on the two parameters $\alpha$ and $\theta$ ($\rho = 1 + \theta$ using (7)). When $\alpha$ or $\theta$ increase, the quality of the solution improves but run times increase as well. Selecting the appropriate values is a balancing act between these two considerations.

We selected the $n = 654$ and $n = 1060$ instances for $p = 20, 40, 60, 80, 100$ and $n = 3038$ instances for $p = 100, 200, 300, 400, 500$. Two thousand starting solutions were generated for each instance totalling 10,000 cases. These solutions were obtained by a constructive heuristic named as START in [13]. This is similar to the drop method which considers initially all potential sites as open and gradually decreases the number of facilities umtil $p$ is reached except that in START from one iteration to the other two well defined facilities with their assigned customers are combined and replaced by a single facility. In this study, a facility is randomly selected for each case and the original LD solution to (3) is found generating a total of 10,000 optimal solutions to LD problems. We then repeated the process (with the same starting solutions and the selected facility $k$) for various values of $\alpha$ and $\theta$ as listed in Table 1 for a total of 36 experiments. The obtained solutions and their corresponding total run times are then recorded for each pair of these values. In Table 1 we summarize the average results over all values of $p$. Here we report the number of times, out of 10,000, that the optimal solution is obtained, the average percentage of the solution above the optimum, and the total run times. The average percentage is surprisingly low even for $\alpha = 0.5$ and $\theta = 0.5$ (i.e., $\rho = 1.5$), which are the smallest values which we tested. For instance, in Table 1, for $\alpha = 0.5$ and $\rho = 1.5$, deviations of 0.0810%, 0.0234% and 0.0061% are reported for $n = 654$, $n = 1060$ and $n = 3038$, respectively.

From the results reported in Table 1 we selected either $\alpha = 0.5$ which is faster or $\alpha = 1$ which provides better results, and opted for $\rho = 4$ for all subsequent experiments.

## 3.2 Hybridization of FIMP with a powerful metaheuristic

In this section, we aim to highlight the impact of FIMP when embedded as part of a metaheuristic. As an example we use one of the most powerful metaheuristic recently developed in [13, 14] known as the COMB heuristic which is readily available to us but any other metaheuristic could also be used for this

Table 1: Sensitivity Analysis on the Values of $\alpha$ and $\theta(\rho = 1 + \theta)$ in FIMP using average results over all values of $p$

| $\alpha$ | $\theta$ | $\theta$ using (6) | | | $\rho = 1 + \theta$ using (7) | | | $\alpha$ | $\theta$ | $\theta$ using (6) | | | $\rho = 1 + \theta$ using (7) | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | (1) | (2) | (3) | (1) | (2) | (3) | | | (1) | (2) | (3) | (1) | (2) | (3) |
| | | $n = 654$ | | | | | | | | $n = 1,060$ | | | | | |
| 0.5 | 0.5 | 0.0870 | 8,481 | 3.47 | 0.0810 | 8,508 | 3.35 | 0.5 | 0.5 | 0.0235 | 9,037 | 5.26 | 0.0234 | 9,032 | 5.12 |
| 0.5 | 1.0 | 0.0845 | 8,496 | 3.65 | 0.0724 | 8,582 | 3.46 | 0.5 | 1.0 | 0.0231 | 9,083 | 6.10 | 0.0225 | 9,093 | 5.44 |
| 0.5 | 1.5 | 0.0822 | 8,519 | 3.91 | 0.0675 | 8,645 | 3.55 | 0.5 | 1.5 | 0.0229 | 9,090 | 7.25 | 0.0216 | 9,126 | 5.82 |
| 0.5 | 2.0 | 0.0800 | 8,531 | 4.16 | 0.0610 | 8,755 | 3.62 | 0.5 | 2.0 | 0.0225 | 9,100 | 8.69 | 0.0205 | 9,140 | 6.40 |
| 0.5 | 2.5 | 0.0789 | 8,545 | 4.55 | 0.0516 | 8,829 | 3.63 | 0.5 | 2.5 | 0.0219 | 9,111 | 10.15 | 0.0192 | 9,158 | 7.03 |
| 0.5 | 3.0 | 0.0775 | 8,561 | 4.72 | 0.0445 | 8,922 | 3.68 | 0.5 | 3.0 | 0.0211 | 9,132 | 11.89 | 0.0182 | 9,178 | 7.73 |
| 1.0 | 0.5 | 0.0316 | 9,223 | 8.77 | 0.0296 | 9,236 | 8.65 | 1.0 | 0.5 | 0.0020 | 9,905 | 25.53 | 0.0020 | 9,899 | 25.19 |
| 1.0 | 1.0 | 0.0313 | 9,229 | 8.90 | 0.0261 | 9,281 | 8.69 | 1.0 | 1.0 | 0.0020 | 9,915 | 26.07 | 0.0020 | 9,915 | 25.43 |
| 1.0 | 1.5 | 0.0304 | 9,242 | 9.26 | 0.0247 | 9,311 | 8.78 | 1.0 | 1.5 | 0.0019 | 9,915 | 26.92 | 0.0019 | 9,915 | 25.68 |
| 1.0 | 2.0 | 0.0297 | 9,246 | 9.30 | 0.0219 | 9,399 | 8.85 | 1.0 | 2.0 | 0.0019 | 9,917 | 27.93 | 0.0018 | 9,915 | 26.03 |
| 1.0 | 2.5 | 0.0296 | 9,249 | 9.67 | 0.0178 | 9,433 | 8.83 | 1.0 | 2.5 | 0.0018 | 9,919 | 29.10 | 0.0017 | 9,915 | 26.45 |
| 1.0 | 3.0 | 0.0292 | 9,250 | 9.87 | 0.0142 | 9,487 | 8.86 | 1.0 | 3.0 | 0.0018 | 9,921 | 30.50 | 0.0016 | 9,916 | 26.98 |
| 1.5 | 0.5 | 0.0017 | 9,950 | 19.02 | 0.0017 | 9,950 | 18.87 | 1.5 | 0.5 | 0.0001 | 9,991 | 49.07 | 0.0001 | 9,990 | 48.61 |
| 1.5 | 1.0 | 0.0017 | 9,952 | 19.16 | 0.0017 | 9,952 | 18.91 | 1.5 | 1.0 | 0.0001 | 9,994 | 49.46 | 0.0001 | 9,994 | 48.70 |
| 1.5 | 1.5 | 0.0017 | 9,952 | 19.35 | 0.0015 | 9,957 | 18.95 | 1.5 | 1.5 | 0.0001 | 9,994 | 49.66 | 0.0001 | 9,994 | 48.74 |
| 1.5 | 2.0 | 0.0017 | 9,952 | 19.39 | 0.0014 | 9,960 | 18.97 | 1.5 | 2.0 | 0.0001 | 9,994 | 50.11 | 0.0001 | 9,994 | 48.95 |
| 1.5 | 2.5 | 0.0017 | 9,952 | 19.72 | 0.0013 | 9,963 | 18.98 | 1.5 | 2.5 | 0.0001 | 9,994 | 50.63 | 0.0001 | 9,994 | 49.09 |
| 1.5 | 3.0 | 0.0017 | 9,952 | 19.96 | 0.0010 | 9,967 | 18.95 | 1.5 | 3.0 | 0.0001 | 9,994 | 51.25 | 0.0001 | 9,994 | 49.23 |
| 2.0 | 0.5 | 0.0002 | 9,990 | 25.90 | 0.0002 | 9,990 | 25.68 | 2.0 | 0.5 | 0 | 10,000 | 60.05 | 0 | 10,000 | 59.51 |
| 2.0 | 1.0 | 0.0002 | 9,990 | 26.12 | 0.0002 | 9,990 | 25.62 | | | $n = 3,038$ | | | | | |
| 2.0 | 1.5 | 0.0002 | 9,990 | 26.51 | 0.0002 | 9,990 | 25.72 | 0.5 | 0.5 | 0.0061 | 8,534 | 46.80 | 0.0061 | 8,523 | 46.75 |
| 2.0 | 2.0 | 0.0002 | 9,990 | 26.30 | 0.0002 | 9,991 | 25.75 | 0.5 | 1.0 | 0.0061 | 8,551 | 47.27 | 0.0061 | 8,548 | 46.95 |
| 2.0 | 2.5 | 0.0002 | 9,990 | 26.33 | 0.0002 | 9,991 | 25.70 | 0.5 | 1.5 | 0.0061 | 8,553 | 48.01 | 0.0060 | 8,553 | 47.28 |
| 2.0 | 3.0 | 0.0002 | 9,990 | 26.69 | 0.0001 | 9,993 | 25.73 | 0.5 | 2.0 | 0.0060 | 8,556 | 49.00 | 0.0060 | 8,559 | 47.60 |
| 2.5 | 0.5 | 0 | 10,000 | 33.10 | 0 | 10,000 | 32.70 | 0.5 | 2.5 | 0.0060 | 8,561 | 50.14 | 0.0058 | 8,563 | 48.08 |
| | | | | | | | | 0.5 | 3.0 | 0.0059 | 8,563 | 51.50 | 0.0057 | 8,566 | 48.63 |
| | | | | | | | | 1.0 | 0.5 | 0.0002 | 9,888 | 321.53 | 0.0002 | 9,884 | 320.89 |
| | | | | | | | | 1.0 | 1.0 | 0.0002 | 9,892 | 321.80 | 0.0002 | 9,892 | 320.85 |
| | | | | | | | | 1.0 | 1.5 | 0.0002 | 9,892 | 322.19 | 0.0002 | 9,892 | 320.66 |
| | | | | | | | | 1.0 | 2.0 | 0.0002 | 9,892 | 322.78 | 0.0002 | 9,893 | 321.20 |
| | | | | | | | | 1.0 | 2.5 | 0.0002 | 9,892 | 323.80 | 0.0002 | 9,893 | 321.43 |
| | | | | | | | | 1.0 | 3.0 | 0.0002 | 9,892 | 324.31 | 0.0002 | 9,894 | 321.75 |
| | | | | | | | | 1.5 | 0.5 | 0 | 10,000 | 556.30 | 0 | 10,000 | 554.83 |

(1) Percent above optimum. (2) # of times optimum obtained. (3) Time (sec.) for all 10,000 runs.

purpose. COMB is a hybrid of a Genetic Algorithm (GA) which is supplemented by an effective variable neighbourhood search (VNS) as a post-optimizer. The GA is proposed by Drezner et al. [14]. The idea behind the merging process of the GA is to draw an imaginary line at a random angle through the center of the cluster of the facilities and to choose the locations of the facilities on one side of the line from one parent and those on the other side of the line from the second parent. The expectation is that if the configuration of each parent on its side of the line is a good one, the merge will produce a superior offspring. The VNS is a distribution based variable neighborhood search (DVNS) algorithm proposed in [14]. This is a variation on the basic VNS algorithm [19, 24]. Traditional VNS algorithms set a parameter $k_{\max}$ and shake the best found solution sequentially in neighborhoods $k = 1, \ldots, k_{\max}$. In DVNS [14], the process generates a random level of shaking $k$ according to a density function $\phi(x)$, for $x \in (0, 1)$, that is highest near some $\frac{k}{k_{\max}}$ and

Table 2: Solution Results by COMB for $n = 654$ Instances

| | Best | IMP | | | VAR1 | | | VAR2 | | | VAR3 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $p$ | Known | (1) | (2) | (3) | (1) | (2) | (3) | (1) | (2) | (3) | (1) | (2) | (3) |
| 20 | 63,389.0238 | 10 | 0 | 2.39 | 10 | 0 | 0.85 | 10 | 0 | 0.89 | 10 | 0 | 0.83 |
| 25 | 52,209.5106 | 10 | 0 | 3.12 | 10 | 0 | 1.00 | 10 | 0 | 1.03 | 10 | 0 | 0.90 |
| 30 | 44,705.1920 | 10 | 0 | 3.89 | 10 | 0 | 1.31 | 10 | 0 | 1.25 | 10 | 0 | 1.13 |
| 35 | 39,257.2685 | 10 | 0 | 5.21 | 10 | 0 | 1.60 | 10 | 0 | 1.53 | 10 | 0 | 1.53 |
| 40 | 35,704.4076 | 10 | 0 | 6.55 | 10 | 0 | 1.98 | 10 | 0 | 1.88 | 10 | 0 | 1.83 |
| 45 | 32,306.9721 | 10 | 0 | 7.93 | 10 | 0 | 2.36 | 10 | 0 | 2.22 | 10 | 0 | 2.14 |
| 50 | 29,338.0106 | 10 | 0 | 9.15 | 10 | 0 | 2.75 | 10 | 0 | 2.63 | 10 | 0 | 2.32 |
| 55 | 26,699.1208 | 10 | 0 | 10.65 | 10 | 0 | 3.00 | 10 | 0 | 2.86 | 10 | 0 | 2.61 |
| 60 | 24,504.3952 | 10 | 0 | 11.96 | 10 | 0 | 3.25 | 10 | 0 | 3.13 | 10 | 0 | 2.90 |
| 65 | 22,733.2923 | 10 | 0 | 13.66 | 10 | 0 | 3.81 | 10 | 0 | 3.68 | 10 | 0 | 3.42 |
| 70 | 21,465.4361 | 10 | 0 | 16.77 | 10 | 0 | 4.20 | 10 | 0 | 4.04 | 10 | 0 | 3.79 |
| 75 | 20,269.9644 | 10 | 0 | 19.00 | 10 | 0 | 4.92 | 10 | 0 | 4.72 | 10 | 0 | 4.47 |
| 80 | 19,193.8610 | 10 | 0 | 20.50 | 10 | 0 | 5.24 | 10 | 0 | 5.04 | 10 | 0 | 4.85 |
| 85 | 18,313.8703 | 10 | 0 | 22.42 | 10 | 0 | 6.10 | 10 | 0 | 5.90 | 10 | 0 | 5.85 |
| 90 | 17,514.4227 | 10 | 0 | 24.25 | 10 | 0 | 7.29 | 10 | 0 | 7.42 | 10 | 0 | 6.56 |
| 95 | 16,770.1973 | 9 | 0.0003 | 27.56 | 6 | 0.0013 | 8.10 | 7 | 0.0010 | 9.12 | 6 | 0.0013 | 7.55 |
| 100 | 16,083.5345 | 10 | 0 | 30.30 | 10 | 0 | 8.48 | 10 | 0 | 8.17 | 10 | 0 | 7.80 |
| Average: | | 9.9 | 0.00002 | 13.84 | 9.8 | 0.00008 | 3.90 | 9.8 | 0.00006 | 3.85 | 9.8 | 0.00008 | 3.56 |

(1) Number of times in 10 runs that BK found. (2) Percentage of average above BK. (3) Time (min.) per run.

lower near 0 and 1. The next $k$ is derived by multiplying $k_{\max}$ by this random value.

In the computational experiments we compare the original COMB that uses IMP with the new implementation of COMB that is based on the following three variants of FIMP.

(i) The basic variant (VAR1):

- Generate 100 starting solutions using $\alpha = 1$,
- Apply the GA using $100p$ generations without improvement (in [14], $\frac{np}{5}$ generations were used instead),
- In DVNS:
  - Use $k_{\max} = 20$
  - Set $\alpha = 0.5$ and use the NB rule with $\rho = 4$.
  - Terminate when $100 \min\{p, 20\}$ iterations are used without improvement.

(ii) VAR2: as VAR1 except that $\alpha = 0.5$ is used in the generation phase.

(iii) VAR3: as VAR1 except that $k_{\max} = 10$ is used in DVNS.

It was observed that the improving perturbations in FIMP very rarely used neighborhoods with $k > 10$. This is why we created and tested VAR3. This way more potentially improving neighborhoods are tested.

In Tables 2 and 3 the instances with $n = 656$ and $n = 1060$ are compared for COMB contrasting using IMP and FIMP. Note that when IMP was applied the number of iterations in DVNS was only $50 \min\{p, 20\}$.

Table 3: Solution Results by COMB for $n = 1060$ Instances

| | Best | IMP | | | VAR1 | | | VAR2 | | | VAR3 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $p$ | Known | (1) | (2) | (3) | (1) | (2) | (3) | (1) | (2) | (3) | (1) | (2) | (3) |
| 5 | 1,851,877.3 | 10 | 0 | 0.71 | 10 | 0 | 0.85 | 10 | 0 | 0.88 | 10 | 0 | 0.86 |
| 10 | 1,249,564.8 | 10 | 0 | 1.93 | 10 | 0 | 1.83 | 10 | 0 | 1.83 | 10 | 0 | 1.80 |
| 15 | 980,131.7 | 10 | 0 | 4.31 | 10 | 0 | 3.25 | 10 | 0 | 3.21 | 10 | 0 | 3.01 |
| 20 | 828,685.7 | 10 | 0 | 6.96 | 10 | 0 | 4.30 | 10 | 0 | 3.77 | 10 | 0 | 3.61 |
| 25 | 721,988.2 | 10 | 0 | 9.12 | 10 | 0 | 4.42 | 10 | 0 | 3.94 | 10 | 0 | 3.76 |
| 30 | 638,212.3 | 10 | 0 | 10.15 | 10 | 0 | 4.29 | 10 | 0 | 3.91 | 10 | 0 | 3.77 |
| 35 | 577,496.7 | 10 | 0 | 11.78 | 10 | 0 | 4.48 | 10 | 0 | 4.11 | 10 | 0 | 4.17 |
| 40 | 529,660.1 | 10 | 0 | 13.94 | 10 | 0 | 4.83 | 10 | 0 | 4.31 | 10 | 0 | 4.55 |
| 45 | 489,483.8 | 10 | 0 | 16.68 | 10 | 0 | 5.72 | 10 | 0 | 5.03 | 10 | 0 | 5.27 |
| 50 | 453,109.6 | 10 | 0 | 18.79 | 10 | 0 | 5.96 | 10 | 0 | 5.67 | 10 | 0 | 5.40 |
| 55 | 422,638.7 | 10 | 0 | 20.88 | 10 | 0 | 6.36 | 10 | 0 | 6.39 | 10 | 0 | 5.73 |
| 60 | 397,674.5 | 10 | 0 | 23.53 | 10 | 0 | 6.94 | 10 | 0 | 7.19 | 10 | 0 | 6.37 |
| 65 | 376,630.3 | 10 | 0 | 26.88 | 10 | 0 | 7.85 | 10 | 0 | 8.17 | 10 | 0 | 7.22 |
| 70 | 357,335.1 | 10 | 0 | 29.53 | 10 | 0 | 8.33 | 10 | 0 | 8.43 | 10 | 0 | 7.74 |
| 75 | 340,123.5 | 10 | 0 | 33.67 | 10 | 0 | 9.03 | 10 | 0 | 8.47 | 10 | 0 | 8.31 |
| 80 | 325,971.3 | 10 | 0 | 35.82 | 10 | 0 | 10.17 | 10 | 0 | 8.69 | 10 | 0 | 9.29 |
| 85 | 313,446.6 | 10 | 0 | 40.16 | 10 | 0 | 12.21 | 10 | 0 | 11.19 | 10 | 0 | 10.86 |
| 90 | 302,479.1 | 10 | 0 | 44.32 | 10 | 0 | 14.16 | 10 | 0 | 13.06 | 10 | 0 | 13.64 |
| 95 | 292,282.6 | 10 | 0 | 48.47 | 10 | 0 | 16.97 | 10 | 0 | 12.97 | 10 | 0 | 15.29 |
| 100 | 282,536.5 | 10 | 0 | 52.29 | 10 | 0 | 18.74 | 10 | 0 | 13.83 | 9 | 0.002 | 15.73 |
| 105 | 273,463.3 | 10 | 0 | 60.74 | 10 | 0 | 20.14 | 10 | 0 | 16.86 | 10 | 0 | 18.26 |
| 110 | 264,959.6 | 10 | 0 | 65.89 | 10 | 0 | 19.85 | 10 | 0 | 18.95 | 9 | 0.000 | 18.84 |
| 115 | 256,735.7 | 9 | 0.001 | 75.99 | 10 | 0 | 24.10 | 10 | 0 | 19.65 | 10 | 0 | 22.77 |
| 120 | 249,050.5 | 8 | 0.001 | 83.93 | 10 | 0 | 24.41 | 10 | 0 | 23.44 | 10 | 0 | 22.45 |
| 125 | 241,880.4 | 10 | 0 | 93.77 | 9 | 0.002 | 32.83 | 9 | 0.002 | 24.84 | 10 | 0 | 26.60 |
| 130 | 235,203.4 | 10 | 0 | 86.45 | 10 | 0 | 30.07 | 10 | 0 | 27.21 | 10 | 0 | 26.38 |
| 135 | 228,999.2 | 5 | 0.001 | 106.24 | 9 | 0.001 | 32.74 | 7 | 0.001 | 31.38 | 8 | 0.001 | 31.29 |
| 140 | 223,062.0 | 9 | 0.001 | 112.42 | 10 | 0 | 36.23 | 8 | 0.004 | 36.51 | 8 | 0.003 | 33.14 |
| 145 | 217,462.8 | 8 | 0.000 | 121.49 | 10 | 0 | 41.03 | 9 | 0.000 | 40.93 | 10 | 0 | 39.07 |
| 150 | 212,230.5 | 9 | 0.000 | 164.00 | 10 | 0 | 49.78 | 10 | 0 | 40.88 | 10 | 0 | 43.80 |
| Average: | | 9.6 | 0.0002 | 47.36 | 9.9 | 0.0001 | 15.40 | 9.8 | 0.0003 | 13.86 | 9.8 | 0.0002 | 13.97 |

(1) Number of times in 10 runs that BK found. (2) Percentage of average above BK. (3) Time (min.) per run.

Run times for these instances by FIMP were cut to about one third of those by IMP with a negligible deterioration in solution quality.

Run times for BVNS or DVNS with IMP were very long for the $n = 3,038$ instances: about 36 days for one run of solving all 10 instances which will require about a year of computer time for completing 10 runs for either BVNS and DVNS. Running COMB will probably require even longer run times. Therefore, only the three variants of COMB using FIMP were tested on the $n = 3,038$ instances. The experiments with the $n = 3,038$ instances still required relatively long run times and thus each instance was run three times on the fast computer and seven more times on other computers. Run times are reported as the average of the three runs on the fast computer for each instance.

In Tables 4 and 5 we report the results for the $n = 3038$ instances. Run times are quite reasonable. Running all ten instances once by either variant required about 8-9 days. The best known solutions to date are reported by Taillard [30] for $p \geq 100$ (results for $p = 50$ are not reported in [30]) and by Brimberg et al. [5] for $p = 50$. New best known (NBK) results were obtained for eight of the ten instances ($p = 50$

Table 4: Results for the $n = 3,038$ Instances

| $p$ | New Best Known (NBK) | VAR1 | | | VAR2 | | | VAR3 | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | (1) | (2) | (3) | (1) | (2) | (3) | (1) | (2) | (3) |
| 50 | 505,875.76 | 10 | 0 | 0.46 | 10 | 0 | 0.38 | 10 | 0 | 0.42 |
| 100 | 351,171.14 | 8 | 0.005 | 1.13 | 9 | 0.002 | 0.96 | 5 | 0.009 | 1.11 |
| 150 | 279,724.73 | 6 | 0.006 | 2.35 | 4 | 0.010 | 2.37 | 4 | 0.007 | 2.62 |
| 200 | 236,209.47 | 3 | 0.002 | 4.78 | 0 | 0.006 | 3.53 | 2 | 0.003 | 4.36 |
| 250 | 206,454.72 | 2 | 0.006 | 8.43 | 0 | 0.013 | 6.65 | 2 | 0.008 | 7.87 |
| 300 | 184,802.33 | 0 | 0.006 | 14.27 | 1 | 0.009 | 10.62 | 0 | 0.006 | 13.76 |
| 350 | 168,254.30 | 1 | 0.006 | 21.17 | 0 | 0.011 | 18.20 | 0 | 0.007 | 19.14 |
| 400 | 154,554.55 | 1 | 0.018 | 34.26 | 0 | 0.016 | 38.13 | 1 | 0.018 | 32.75 |
| 450 | 143,272.78 | 0 | 0.016 | 50.03 | 1 | 0.017 | 51.42 | 0 | 0.013 | 49.80 |
| 500 | 133,555.40 | 0 | 0.020 | 69.86 | 0 | 0.017 | 87.14 | 1 | 0.016 | 65.65 |
| Average: | | 3.1 | 0.0084 | 20.68 | 2.5 | 0.0101 | 21.94 | 2.5 | 0.0087 | 19.75 |

(1) Number of times out of 10 runs that NBK found.
(2) Percentage of average above NBK. (3) Time (hours) per run

Table 5: % of Best Result Above NBK ($n = 3,038$ Instances)

| $p$ | [5] | [30] | VAR1 | VAR2 | VAR3 |
|---|---|---|---|---|---|
| 200 | 0.275 | 0.036 | 0 | 0.000 | 0 |
| 250 | 0.423 | 0.035 | 0 | 0.001 | 0 |
| 300 | 0.421 | 0.017 | 0.002 | 0 | 0.001 |
| 350 | 0.463 | 0.042 | 0 | 0.006 | 0.000 |
| 400 | 0.539 | 0.067 | 0 | 0.004 | 0 |
| 450 | 0.406 | 0.040 | 0.004 | 0 | 0.003 |
| 500 | 0.361 | 0.027 | 0.008 | 0.006 | 0 |
| Max | 0.539 | 0.067 | 0.008 | 0.006 | 0.003 |

and all $200 \leq p \leq 500$ instances). The same best known results were obtained for the other two instances ($p = 100, 150$).

It seems that VAR2 is slightly inferior to the other two variants. VAR3 performed best for large values of $p$ but did not perform as well as the other two variants for small values of $p$

**Statistical Analysis**

We compared the results for $p = 450$ and 500 by VAR1 and VAR3. Since the first two phases of COMB are the same for these two variants, we performed a comparison test to see whether or not $k_{\max} = 10$ is better than $k_{\max} = 20$ for these instances. We ran a paired $t$-test on these 20 observations for each variant. It was found that VAR3 ($k_{\max} = 10$) provided better results than VAR1 ($k_{\max} = 20$) with a p-value of 0.014.

We also estimated run times for $n = 3038$ instances by multiple regression. We assumed the relationship $t(p) = \alpha p^{\beta+\gamma p}$. Since there are three variants we assume that each one has a different value of $\alpha$ but the same values of $\beta$ and $\gamma$. This yields a regression model similar to the idea of dummy variables in forecasting. Let $t(p, i)$ be the run time for instance $p$ and VAR$i$ for $i = 1, 2, 3$. For VAR1 the coefficient is $\alpha$ but for VAR2 and VAR3 the coefficients are $\alpha\theta_2$ and $\alpha\theta_3$, respectively. By taking the logarithm of the equation we

get $\log(t(p,i)) = \log\alpha + u_2 \log\theta_2 + u_3 \log\theta_3 + \beta \log p + \gamma p \log p$ where for VAR1 $u_2 = u_3 = 0$, for VAR2 $u_2 = 1$ and $u_3 = 0$ and for VAR3 $u_2 = 0$ and $u_3 = 1$. The data consists of 30 rows each with a dependent variable $\log(t(p,i))$ and four independent variables $\log p$, $p \log p$, $u_2$ and $u_3$. The significance of the regression is $10^{-30}$ and the first three variables are $\alpha = 0.003742$, $\beta = 1.163$, $\gamma = 0.000889$ each with a p-value of about $10^{-14}$. The two dummy variables were factor of 0.906 for VAR2 (p-value=0.04) and 0.96 for VAR3 which is not significant. It is interesting to note that run times for VAR2 seem to grow faster for large values of $p$. Even though the average run time for VAR2 is greater than that of VAR1, the regression indicates that run times are about 90% of those for VAR1.

# 4 Conclusions

Two simple but effective neighborhood reduction rules are developed and embedded within a well known local search to solve the planar Euclidean $p$-median problem. Massive reduction of CPU time, in some cases 90% saving, was obtained at an expense of a negligible loss in solution quality. This fast local search is then embedded into a powerful metaheuristic algorithm for illustration purposes. Computational experiments on a set of well researched test problems provided excellent results including new best results. This study empirically demonstrates the need for considering neighborhood reduction schemes as an integral part of heuristic search design for global optimisation in general.

# References


[1] Berman, O., Drezner, Z., and Krass, D. (2011). Big segment small segment global optimization algorithm on networks. *Networks*, 58:1–11.

[2] Bongartz, I., Calamai, P. H., and Conn, A. R. (1994). A projection method for $\ell_p$ norm location-allocation problems. *Mathematical Programming*, 66:238–312.

[3] Brimberg, J. and Drezner, Z. (2013). A new heuristic for solving the p-median problem in the plane. *Computers & Operations Research*, 40:427–437.

[4] Brimberg, J., Drezner, Z., Mladenović, N., and Salhi, S. (2014). A new local search for continuous location problems. *European Journal of Operational Research*, 232:256–265.

[5] Brimberg, J., Hansen, P., and Mladenović, N. (2006). Decomposition strategies for large-scale continuous location–allocation problems. *IMA Journal of Management Mathematics*, 17:307–316.

[6] Brimberg, J., Hansen, P., Mladenović, N., and Salhi, S. (2008). A survey of solution methods for the continuous location-allocation problem. *International Journal of Operations Research*, 5:1–12.

[7] Brimberg, J., Hansen, P., Mladenović, N., and Taillard, E. (2000). Improvements and comparison of heuristics for solving the uncapacitated multisource Weber problem. *Operations Research*, 48:444–460.

[8] Chen, P. C., Hansen, P., Jaumard, B., and Tuy, H. (1998). A fast algorithm for the greedy interchange for large-scale clustering and median location problems by D.-C. programming. *Operations Research*, 46:548–562.

[9] Chen, R. (1983). Solution of minisum and minimax location-allocation problems with euclidean distances. *Naval Research Logistics Quarterly*, 30:449–459.

[10] Cooper, L. (1963). Location-allocation problems. *Operations Research*, 11:331–343.

[11] Cooper, L. (1964). Heuristic methods for location-allocation problems. *SIAM Review*, 6:37–53.

[12] Drezner, Z. (1984). The planar two-center and two-median problems. *Transportation Science*, 18:351–361.

[13] Drezner, Z., Brimberg, J., Salhi, S., and Mladenović, N. (2014). New local searches for solving the multi-source Weber problem. *Annals of Operations Research (in review)*.

[14] Drezner, Z., Brimberg, J., Salhi, S., and Mladenović, N. (2014). New heuristic algorithms for solving the planar $p$-median problem. *Computers and Operations Research*. in press.

[15] Drezner, Z., Mehrez, A., and Wesolowsky, G. O. (1991). The facility location problem with limited distances. *Transportation Science*, 25:183–187.

[16] Drezner, Z. and Suzuki, A. (2004). The big triangle small triangle method for the solution of non-convex facility location problems. *Operations Research*, 52:128–135.

[17] Eilon, S., Watson-Gandy, C. D. T., and Christofides, N. (1971). *Distribution Management*. Hafner, New York.

[18] Golden, B. L., Magnanti, T. L., and Nguyen, H. Q. (1977). Implementing vehicle routing algorithms. *Networks*, 7:113–148.

[19] Hansen, P. and Mladenović, N. (1997). Variable neighborhood search for the $p$-median. *Location Science*, 5:207–226.

[20] Hansen, P., Peeters, D., and Thisse, J.-F. (1981). On the location of an obnoxious facility. *Sistemi Urbani*, 3:299–317.

[21] Krau, S. (1997). *Extensions du problème de Weber*. PhD thesis, École Polytechnique de Montréal.

[22] Megiddo, N. and Supowit, K. J. (1984). On the complexity of some common geometric location problems. *SIAM Journal on Computing*, 13:182–196.

[23] Mladenović, N., Dražić, M., Kovačevic-Vujčić, V., and Čangalović, M. (2008). General variable neighborhood search for the continuous optimization. *European Journal of Operational Research*, 191(3):753–770.

[24] Mladenović, N. and Hansen, P. (1997). Variable neighborhood search. *Computers & Operations Research*, 24:1097–1100.

[25] Murtagh, B. A. and Niwattisyawong, S. R. (1982). An efficient method for the multi-depot location-allocation problem. *Journal of the Operational Research Society*, 33:629–634.

[26] Plastria, F. (1992). GBSSS, the generalized big square small square method for planar single facility location. *European Journal of Operational Research*, 62:163–174.

[27] Reinelt, G. (1991). TSLIB a traveling salesman library. *ORSA Journal on Computing*, 3:376–384.

[28] Salhi, S. and Sari, M. (1997). A multi-level composite heuristic for the multi-depot vehicle fleet mix problem. *European Journal of Operational Research*, 103:95–112.

[29] Schöbel, A. and Scholz, D. (2010). The big cube small cube solution method for multidimensional facility location problems. *Computers and Operations Research*, 37:115–122.

[30] Taillard, É. (2003). Heuristic methods for large centroid clustering problems. *Journal of Heuristics*, 9:51–73.