New Heuristic Algorithms for Solving 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

Jack Brimberg
Department of Mathematics and Computer Science
The Royal Military College of Canada
Kingston, ON Canada.
e-mail: Jack.Brimberg@rmc.ca

Nenad Mladenović
LAMIH Laboratory
University of Valenciennes and Hainaut-Cambrsis Universit de Valenciennes
Le Mont Houy, 59313, France.

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

In this paper we propose effective heuristics for the solution of the planar p-median problem. We develop a new distribution based variable neighborhood search and a new genetic algorithm, and also test a hybrid algorithm that combines these two approaches. The best results were obtained by the hybrid approach. The best known solution was found in 466 out of 470 runs, and the average solution was only 0.000016% above the best known solution on 47 well explored test instances of 654 and 1060 demand points and up to 150 facilities.

Key Words: Location Analysis; Planar p-median; Multi-source Weber Problem; Variable Neighborhood Search; Genetic Algorithm.

*This research has been supported in part by a Natural Sciences and Engineering Research Council of Canada Discovery Grant (NSERC #205041-2008) and by the UK Research Council EPSRC(EP/I009299/1). The third author is partly supported by Project number 144010 funded by the Serbiam Ministery of Sciences.

†Corresponding Author

1 Introduction

The location-allocation problem in the plane, also known as 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 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:

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

Drezner [18] and Chen et al. [12] developed optimal solution procedures for the location of $p = 2$ facilities. Schöbel and Scholz [39] optimally solved problems with $p = 2, 3$ facilities. Krau [31] used column generation and optimally solved problems with $n = 50$ and 287 demand points and any number of facilities. This method is highly sensitive to the quality of the starting solution.

The continuous p-median problem (1) is known to be NP-hard [32], and as a result, many heuristics have been developed to solve it. Classical heuristics include the famous alternating procedure by Cooper [14, 15], the projection method of Bongartz et al. [2], and gradient-based methods such as [35, 13]. Brimberg and Drezner [3] proposed several heuristics for solving the p-median problem. One of the approaches is IALT which is a modification of Cooper's alternate algorithm [14, 15]. Brimberg et al. [5] developed the reformulation local search (RLS) which is a new local search that iterates between the continuous problem and a discrete approximation. Drezner et al. [22] also proposed a constructive algorithm START, a decomposition approach, and a local search IMP that provide better results than the Cooper like approaches [3, 14, 15]. The leading heuristics to date for solving the planar p-median problem are based on variable neighborhood search (e.g. see [8, 7]). Decomposition strategies apply well to larger scale problem instances [6, 40]. Further gains may be obtained by using new local searches (e.g., [22]) or variable neighborhood descent within a general variable neighborhood search [33]. For recent reviews of solution approaches to the continuous p-median problem the reader is referred to [8, 7, 5, 22].

The following algorithm abbreviations are used in this paper.

ALT: Cooper's alternate algorithm [14, 15].

BVNS: The basic variable neighborhood search developed in this paper.

COMB: The combined approach developed in this paper.

DVNS: The distribution based variable neighborhood search developed in this paper.

GA: The genetic algorithm developed in this paper.

IALT: A modification of Cooper's alternate algorithm [3].

IDEC3: A decomposition-based heuristic [22].

IMP: A local search based on LD [22]. Used in all the heuristics developed in this paper.

LD: The limited distance location problem [24].

START: A constructive algorithm to generate starting solutions [22].

VNS: The variable neighborhood search [29, 34].

The contributions of the present paper relate to the construction of new metaheuristic-based algorithms. The IALT, START and IMP procedures were proposed in earlier papers [3, 4, 22]. The decomposition-based local search developed in [22] and the reformulation based local search in [5] are not utilized in this paper. We designed (i) A basic VNS algorithm using IMP as the local search (ii) a new distribution based VNS (DVNS) which is a modification of the basic VNS, (iii) a genetic algorithm (GA), and (iv) a combined approach of both GA and DVNS which provided the best results. We also note that these algorithms are readily adapted to other continuous location problems besides the planar p-median problem.

For completeness we review the START and IMP algorithms in the next section. Section 3 describes the basic variable neighborhood search with IMP in the local search step. We also develop a new approach which we term "distribution based" variable neighborhood search (DVNS). In this approach the shaking operation selects the next neighborhood according to a prescribed probability distribution. Comparable results are obtained as in basic VNS but in half the time. A new genetic algorithm is proposed as well that uses a simple and effective geometric merging process. We conclude Section 3 with setting parameter values for the various heuristics. The computational

experiments are discussed in Section 4, followed by conclusions and suggestions for future research.

2 Procedures Applied in this Paper

In this section We briefly describe the existing procedures START [22] and IMP [22] applied in this paper.

2.1 The START Algorithm

The outline of START is as follows. For complete details the reader is referred to [22]. The main idea is to form p initial clusters of the demand points in a greedy constructive way.

Each demand point initially belongs to its own subset defining n facilities. Set $v_i = w_i$ for $i = 1, \ldots, n$. Calculate for all pairs $i < j$

$$\Delta_{ij} = \frac{v_i v_j}{v_i + v_j} d_{ij}(\theta + u) \tag{2}$$

where d_{ij} is the distance between current facilities i and j, and u is a random variable in $[0, 1]$ that is used as a stochastic perturbation. In our experiments we used $\theta = 0.25$.

Repeat the following until the number of subsets is reduced to p.

1. Find the pair $i < j$ for which Δ_{ij} is minimized.
2. Create a location for a new facility at $\frac{v_i X_i + v_j X_j}{v_i + v_j}$ with a weight $v_i + v_j$.
3. Remove current facilities i and j, and label the index of the new facility and its weight with i. Calculate for all $r \neq i$: Δ_{ri} for $r < i$ and Δ_{ir} for $r > i$ by (2). The number of facilities is reduced by one.

The START algorithm uses the binary heap data structure ([10, 28]) for faster execution. Once the p clusters are formed, each defines a facility and the solution is then improved by the IALT algorithm [3].

2.2 The IMP Algorithm

The IMP algorithm was proposed and detailed in [22]. For completeness we describe it briefly here. Suppose that the 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 present locations can be found as follows. The shortest distance between demand point i and the p facilities is the minimum between the unknown distance to facility k and the minimum distance to all other facilities which is fixed and does not depend on the location of facility k. Define the minimum distance to the fixed facilities as

$$D_i = \min_{j \neq k} \{d_i(X_j)\}. \tag{3}$$

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

$$G(X_k) = \sum_{i=1}^{n} w_i \min \{d_i(X_k), D_i\}. \tag{4}$$

where D_i are constants defined by (3).

This is the limited distance location problem (LD) that can be formulated in general for any number of facilities. We apply the single facility version of the problem introduced in [24], which can be optimally solved by global optimization techniques such as Big Square Small Square (BSSS) proposed by Hansen et al. [30] and improved by Plastria [36]. The complete solution procedure is given in [22]. Problems with up to 100,000 demand points are optimally solved in less than 3 seconds of computing time.

The steps of the algorithm are outlined below.

1. Obtain a starting solution by START [22] or any other method.
2. Set $index = 0$.
3. Repeat the following for each facility k in random order.
   (a) Calculate D_i by (3) for $i = 1, \ldots, n$.
   (b) Relocate facility k by solving the LD problem (4).
   (c) If the location of facility k changed, set $index = 1$.

4. If $index = 1$, return to Step 2; else stop (the current solution is a local optimum).

Note that in Step 3b the value of the objective function (1) must be at least as good as the present value of the objective function because the algorithm finds the optimal solution of LD for facility k. Also note that in Cooper-style alternating locate-allocate heuristics it is possible to obtain a solution where a facility has no demand points assigned to it. This degeneracy problem, discussed in Brimberg and Mladenović [9], does not occur in the IMP approach.

In our experiments the IMP algorithm provided better results than the Cooper-style alternate algorithm ALT or IALT [14, 15, 3]. There are two main reasons for the excellent performance of IMP.

1. When facility k is relocated, the ALT or IALT algorithms hold the set S of demand points serviced by facility k fixed while LD may add or remove demand points from S. When the location of facility k is changed, demand points near the periphery of S may become closer to a fixed facility and thus removed from S, while points not in S may become closer to facility k and thus added to S.

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), locating the facility in an "optimal" location in a different region of the solution space may decrease the value of the objective function more than the increase following its removal. This may be quite common in VNS algorithms when the perturbed facility is "close" to another facility and thus one of them may be relocated to another region of the solution space.

3 New Metaheuristic Algorithms

Four metaheuristic algorithms are proposed. The first one is the basic variable neighborhood search (BVNS) [29, 34] using the powerful IMP local search [22]. We then modified BVNS to a distribution based VNS (DVNS) by selecting the shaking in random order rather than sequentially. The third metaheursitc is a genetic algorithm (GA) based on an effective merging process for producing

offsprings. We then also tested a combined approach (COMB) applying the distribution based variable neighborhood search on the solution of the genetic algorithm.

3.1 The Basic Variable Neighborhood Search Algorithm

The basic VNS (BVNS) algorithm is designed as follows. A starting solution is given and serves as the current solution.

A perturbation is defined as selecting a facility at random and moving it to a random point. In keeping with current practice (e.g., see [7]), we simplify the procedure by restricting the relocation of the facility to a random demand point. Alternative procedures consider the set of grid points obtained by drawing horizontal and vertical lines through the demand points as the set of relocation sites [8].

The k^{th} neighborhood of the current solution is the result of performing a sequence of k perturbations. A facility or a demand point may be selected at most once during the sequence.

The neighborhoods are tested in order for $1 \leq k \leq k_{\max}$. For each k a perturbed solution is randomly generated in the k^{th} neighborhood and subjected to the powerful local search IMP [22]. An iteration is completed when an improved solution occurs or the $k_{\max}$ neighborhood is reached, whichever comes first. Once an improved solution is found, it replaces the current solution and the whole process restarts. The algorithm terminates when the current solution (the original starting solution or a replacement) is not improved in K successive iterations. The procedure requires two parameters: $k_{\max}$ and K. K controls the run time of the algorithm. Larger values of K provide better solutions at the expense of a longer computing time.

The BVNS Algorithm

1. Input an initial solution X_c.
2. Set $k = 1$, $j = 1$.
3. Select a solution X at random in the k^{th}-neighborhood of X_c (termed shaking).
4. Apply IMP [22] on X to obtain a local minimum X_L.

5. If X_L is a better solution than X_c, set $X_c = X_L$, and go to Step 2.
6. Set $k = k + 1$. If $k \leq k_{\max}$, return to Step 3.
7. Set $j = j + 1$. If $j \leq K$, set $k = 1$ and return to Step 3, else stop (final solution is X_c).

3.2 The Distribution Based VNS Algorithm (DVNS)

Traditional VNS algorithms set a parameter $k_{\max}$ and shake the best found solution sequentially in neighborhoods $k = 1, \ldots, k_{\max}$. Once a better solution is found, k is reset to 1. This results in more shakes for $k = 1$ and a declining number for higher values of k. Thus, the total number of shakes to a neighborhood tends to decrease with k. We therefore propose to apply shakes in approximate proportion to the distribution of neighborhoods that result in improved solutions. Thus, successive neighborhoods are generated at random according to a specified distribution, instead of the sequential pattern above. We term this approach as "distribution based" VNS. Similar ideas are proposed in Xiao et al. [41], Carrizosa et al. [11], and Das and Suganthan [16].

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

3.2.1 Analysis

Suppose we wish the mode of the density function $\phi(x)$ to be at some value $m \in [0, 1]$. We look for a transformation function $\psi(x)$ so that when x is randomly generated by a uniform distribution in $[0, 1]$, $\psi(x)$ is a random variable with a density function $\phi(x)$. The density function $\phi(x)$ of selecting a value x is proportional to $\frac{1}{\psi'(x)}$. We assume that $\phi(x) = \frac{1}{\alpha(x-m)^2+\beta}$ for some values of α and β. This leads to $\psi'(x) = \alpha(x - m)^2 + \beta$. We need the transformation $\psi(x)$ to satisfy $\psi(0) = 0; \psi(1) = 1$. Adding a parameter θ, the following transformation function satisfies these three conditions:

$$\psi(x) = \frac{x}{(1 - \frac{3}{2}m)^2 + \theta} \left[(x - \frac{3}{2}m)^2 + \theta \right] \tag{5}$$

For a large enough θ, the function $\psi(x)$ is monotonically increasing for $0 \leq x \leq 1$ and satisfies $\psi(0) = 0; \psi(1) = 1$. The density function is proportional to

$$\frac{1}{\psi'(x)} = \frac{(1 - \frac{3}{2}m)^2 + \theta}{\frac{9}{4}m^2 + \theta - 3x(2m - x)} = \frac{(1 - \frac{3}{2}m)^2 + \theta}{3(x - m)^2 + \theta - \frac{3}{4}m^2}. \tag{6}$$

Integrating to normalize the density function so that its integral is equal to 1 leads to:

$$\phi(x) = \frac{\sqrt{3(\theta - \frac{3}{4}m^2)}}{\left[3(x - m)^2 + \theta - \frac{3}{4}m^2\right]\left[\arctan\sqrt{\frac{3(1-m)^2}{\theta - \frac{3}{4}m^2}} + \arctan\sqrt{\frac{3m^2}{\theta - \frac{3}{4}m^2}}\right]} \tag{7}$$

The density function is valid (positive) when $\theta > \frac{3}{4}m^2$.

It is more convenient to specify $\lambda = \frac{\phi(m)}{\phi(0)} > 1$ rather than θ. By (7)

$$\lambda = \frac{\theta + \frac{9}{4}m^2}{\theta + \frac{3}{4}m^2}$$

leading to

$$\theta = \frac{3}{4}m^2\frac{\lambda + 3}{\lambda - 1} \tag{8}$$

$$\psi(x) = \frac{x}{1 - 3m + \frac{3\lambda}{\lambda-1}m^2}\left[x^2 - 3mx + \frac{3\lambda}{\lambda - 1}m^2\right] \tag{9}$$

$$\phi(x) = \frac{\frac{m}{\sqrt{\lambda-1}}}{\left[(x - m)^2 + \frac{m^2}{\lambda-1}\right]\left[\arctan\left(\frac{1-m}{m}\sqrt{\lambda - 1}\right) + \arctan\sqrt{\lambda - 1}\right]} \tag{10}$$

Note that in the limit when $\lambda \to 1^+$, $\psi(x) = x$ and $\phi(x) = 1$ (a uniform density function).

The DVNS Algorithm

The parameters $k_{\max}$ and K are given. We have flexibility in selecting the mode m of the density function and λ which is the ratio between the maximum value of the density function (at m) and its value at 0. Note that it is much easier to generate a random variable having the density function $\phi(x)$ given in (10) by generating x uniformly in (0, 1) and transforming it by $\psi(x)$ taken from (9).

1. A starting solution X_C is given.
2. Set $j = 1$.

3. Generate uniformly a random number $u \in (0,1)$ and calculate $k = \lfloor \psi(u) k_{\max} \rfloor + 1$ using (9).
4. Randomly select a member of the k^{th} neighborhood and improve it by IMP.
5. If the solution is better than X_C, update X_C and return to Step 2.
6. Set $j = j + 1$.
7. If $j \leq K \times k_{\max}$ go to Step 3.
8. Otherwise, stop with the solution X_C.

To have a fair comparison between the BVNS and DVNS algorithms, the K applied in DVNS should be $k_{\max}$ times the K used in BVNS.

3.3 The Genetic Algorithm (GA)

Salhi and Gamal [38] constructed a genetic algorithm for solving the planar p-median problem. We propose a much simpler genetic algorithm whose success can be attributed to an effective generation of offsprings.

A population of *pop* solutions is maintained. For each population member p facility locations and the value of the objective function are saved. The algorithm is run until G generations do not improve the best known solution.

1. A starting population is generated by running IMP *pop* times. The best and worst values of the objective function among the population members, $f_{\min}$ and $f_{\max}$, are established.
2. Set the generation counter $g = 0$.
3. Set $g = g + 1$. If $g > G$, stop with $f_{\min}$ as the solution.
4. Two parents are randomly selected.
5. An offspring is created by merging the two parents and its objective value f is calculated.
6. If $f \geq f_{\max}$ or f is equal to a population member objective, go to Step 3.
7. The offspring replaces the worst population member and $f_{\max}$ is updated.
8. If $f < f_{\min}$, update $f_{\min} = f$ and go to Step 2.
9. Otherwise, go to Step 3.

The success of the genetic algorithm usually depends on the quality of the merging process. The following merging process exploits the geometry of the problem and is similar to the merging processes suggested in [25, 19] for the solution of other problems.

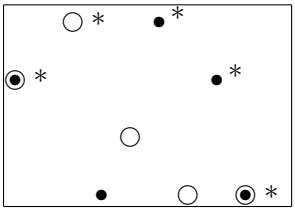
The Merging Process

The idea 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 the configuration of each parent on its side of the line is a good one and the merge will produce a superior offspring. To operationalize it, we need to make sure that the total number of selected facilities is p and each parent contributes about half the facilities. The scheme we came up with is as follows. The local search IALT used in Step 5 refers to the improved Cooper-style alternating method with transfer follow-up given in [3].

1. Randomly generate $\theta \in [0, 2\pi]$.
2. Calculate $v_i = x_i \cos\theta + y_i \sin\theta$ for the locations of each parent. This rotates the axes and facilities by θ and v_i is the projection of facility i on the rotated x-axis.
3. Sort the vector $\{v_i\}$ for $i = 1, \ldots, p$ for each parent.
4. Select the smallest $p_2 = \lfloor \frac{p}{2} \rfloor$ facilities from the first parent and the largest $p - p_2$ facilities from the second parent.
5. Apply IALT on the original locations of the selected set of p facilities.

An illustration of the merging process is depicted in Figure 1. The number of facilities is $p = 5$ and the order of the facilities is irrelevant. Two of the facilities are located at the same point and the other three of each parent are located at different points. The figure depicts the location of the facilities after all ten are rotated by θ. The merged solution consists of the two ($\lfloor \frac{5}{2} \rfloor$) leftmost facilities of the first parent, and the three rightmost facilities of the second parent. The merged solution is marked with a "*".

Figure 1: The two parents following rotation by θ



○ first parent; • second parent;
* merged solution.

3.4 The Combined Approach (COMB)

In this hybrid heuristic we apply DVNS on the final solution of the genetic algorithm. Rather than selecting the best of 100 IMP solutions as a starting solution for DVNS, we apply the genetic algorithm on these 100 IMP solutions as a starting population and select the best solution following the genetic process as a starting solution for DVNS. We selected DVNS rather than BVNS because it performed slightly better in a shorter run time.

3.5 Parameters for the VNS Algorithms

The BVNS and DVNS procedures require two parameters: $k_{\max}$ and K. We applied $k_{\max} = \min\{p, 20\}$, and $K = 50$. For applying DVNS we found that $m = 0.2$ and $\lambda = 2$ approximate the distribution values of k leading to an improved solution. The resulting transformation in (9) is:

$$\psi(x) = \frac{x}{16}\left[25x^2 - 15x + 6\right] \tag{11}$$

with a density function (10):

$$\phi(x) = \frac{1}{[5x^2 - 2x + 0.4]\left[\frac{\pi}{4} + \arctan 4\right]}. \tag{12}$$

This density function is depicted in Figure 2.

Figure 2: The density function for $m = 0.2$, $\lambda = 2$

When $k_{\max}$ is used, the probabilities that $1 \leq k \leq k_{\max}$ is selected is

$$Pr(k) = \int_{\frac{k-1}{k_{\max}}}^{\frac{k}{k_{\max}}} \phi(x)dx. \tag{13}$$

The probabilities for $k_{\max} = 20$ are given in Table 1.

Table 1: Probabilities (in percentages) of selecting the k^{th} neighborhood for $m = 0.2$, $\lambda = 2$

k	$Pr(k)$	k	$Pr(k)$	k	$Pr(k)$	k	$Pr(k)$
1	6.7	6	10.4	11	3.3	16	1.3
2	8.5	7	8.5	12	2.6	17	1.1
3	10.4	8	6.7	13	2.2	18	1.0
4	11.6	9	5.2	14	1.8	19	0.8
5	11.6	10	4.1	15	1.5	20	0.7

Note that small changes in the values of m and λ do not change the probabilities in Table 1 by much. Therefore, the DVNS algorithm performance is not expected to change much if other values of m and λ are selected. For example, we also depict the density function and the probabilities for $m = 0.1$, $\lambda = 1.5$ in Figure 3 and Table 2.

Figure 3: The density function for $m = 0.1$, $\lambda = 1.5$

Table 2: Probabilities (in percentages) of selecting the k^{th} neighborhood for $m = 0.1$, $\lambda = 1.5$

k	$Pr(k)$	k	$Pr(k)$	k	$Pr(k)$	k	$Pr(k)$
1	13.6	6	6.9	11	1.7	16	0.7
2	16.7	7	5.0	12	1.4	17	0.6
3	16.7	8	3.7	13	1.2	18	0.6
4	13.6	9	2.8	14	1.0	19	0.5
5	9.8	10	2.2	15	0.8	20	0.4

3.6 Parameters for the Genetic Algorithm

For the genetic algorithm we used $pop = 100$, $G = \frac{np}{5}$. We selected $pop = 100$ because we selected the best of 100 IMP solutions as the starting solution for the other algorithms. Therefore, the starting solution for GA is comparable to the starting solutions of the other procedures. We also tested $G = 100p$ which replaces $\frac{n}{5}$ with 100. This results in slightly lower solution quality but a reduced run time. Details are reported at the end of the computational experiments section that follows.

4 Computational Experiments

Programs were compiled by an Intel 11.1 Fortran Compiler with no parallel processing and run on a desktop with an Intel 870/i7 2.93GHz CPU Quad processor and 8GB RAM. Only one thread was used.

We used the method in [21] for solving the single facility Weber problem which is faster than other approaches. Even though other operations required a high proportion of the run time, it still saved about 2%-4% of the run time of the genetic algorithm because IALT requires many such solutions. The overwhelming majority of the run time of the other algorithms is taken by IMP which does not require solutions of the single facility Weber problem, and thus their run times are not impacted by the method in [21].

We tested the algorithms on planar p-median problems based on four sets of demand points given in [8] for various values of p for a total of 111 instances. The first set ($n = 50$) is the well-studied 50-customer problem from [26]. The second one ($n = 287$) is the ambulance problem from Bongartz et al. [2]; the last two ($n = 654$, 1060) are taken from the TSP library [37]. Krau [31] found the optimal solution for all the instances of the first two sets. As reported by Drezner et al. [22], the best solution of 100 applications of IMP (repeated 100 times for a total of 10,000 applications for each instance) gave the optimal solution for all these instances, and the best known solution for all $p \leq 15$ for the $n = 654$ instances in all 100 repetitions. Since the starting solution for the algorithms tested in this paper is the best result of 100 applications of IMP, we did not test these instances. We did test and report on the results for the remaining 47 instances.

Drezner et al. [22] report 10 new best known solutions, thus improving solutions reported in [8, 40, 3]. We further improve two of them here, using both BVNS and DVNS, for the $n = 654, p = 95$ and $n = 1060, p = 150$ instances. The new best known solutions are marked in boldface in Table 3.

Comparative results for the $n = 654, 1060$ instances are reported in Table 3. Run times are depicted in Figure 4 and the average run times are reported in Table 4. We first report the starting solutions as a result of running IMP 100 times and selecting the best solution of these 100 runs. We also report results of solutions obtained with IDEC3 (repeated 100 times), a decomposition-

Table 3: Solution Results for $n = 654$ and $n = 1060$ Instances

		Best	100 IMPs		IDEC3		BVNS		DVNS		GA		COMB	
n	p	Known	(1)	(2)	(1)	(2)	(1)	(2)	(1)	(2)	(1)	(2)	(1)	(2)
654	20	63,389.0238	43	0.000	100	0	10	0	10	0	10	0	10	0
654	25	52,209.5106	11	0.001	100	0	10	0	10	0	10	0	10	0
654	30	44,705.1920	9	0.002	82	0.000	10	0	10	0	10	0	10	0
654	35	39,257.2685	3	0.009	58	0.001	10	0	10	0	10	0	10	0
654	40	35,704.4076	0	0.073	25	0.027	10	0	10	0	10	0	10	0
654	45	32,306.9721	0	0.190	42	0.038	10	0	10	0	9	0.000	10	0
654	50	29,338.0106	0	0.238	30	0.073	10	0	10	0	10	0	10	0
654	55	26,699.1208	0	0.189	52	0.031	10	0	7	0.005	10	0	10	0
654	60	24,504.3952	2	0.117	78	0.021	10	0	10	0	10	0	10	0
654	65	22,733.2923	1	0.156	87	0.009	10	0	10	0	10	0	10	0
654	70	21,465.4361	30	0.042	54	0.013	10	0	10	0	10	0	10	0
654	75	20,269.9644	1	0.167	2	0.091	9	0.000	9	0.000	10	0	10	0
654	80	19,193.8610	0	0.365	54	0.072	10	0	10	0	10	0	10	0
654	85	18,313.8703	0	0.436	5	0.119	10	0	9	0.000	10	0	10	0
654	90	17,514.4227	0	0.413	3	0.176	8	0.011	9	0.002	10	0	10	0
654	95	**16,770.1973**	0	0.330	0	0.183	2	0.003	2	0.003	10	0	10	0
654	100	16,083.5345	1	0.211	12	0.105	10	0	10	0	10	0	10	0
1060	5	1,851,877.3	90	0.000	100	0	10	0	10	0	10	0	10	0
1060	10	1,249,564.8	80	0.000	100	0	10	0	10	0	10	0	10	0
1060	15	980,131.7	35	0.002	100	0	10	0	10	0	10	0	10	0
1060	20	828,685.7	3	0.014	100	0	10	0	10	0	10	0	10	0
1060	25	721,988.2	1	0.017	54	0.003	9	0.001	10	0	10	0	10	0
1060	30	638,212.3	17	0.003	100	0	10	0	10	0	10	0	10	0
1060	35	577,496.7	53	0.033	100	0	10	0	10	0	10	0	10	0
1060	40	529,660.1	1	0.109	62	0.024	10	0	10	0	10	0	10	0
1060	45	489,483.8	0	0.185	38	0.056	9	0.001	10	0	10	0	10	0
1060	50	453,109.6	0	0.146	28	0.027	10	0	10	0	10	0	10	0
1060	55	422,638.7	0	0.071	64	0.024	10	0	10	0	10	0	10	0
1060	60	397,674.5	0	0.050	41	0.016	10	0	10	0	10	0	10	0
1060	65	376,630.3	1	0.065	51	0.029	10	0	10	0	10	0	10	0
1060	70	357,335.1	0	0.087	57	0.032	10	0	10	0	10	0	10	0
1060	75	340,123.5	4	0.027	97	0.001	10	0	10	0	10	0	10	0
1060	80	325,971.3	0	0.033	58	0.004	10	0	10	0	10	0	10	0
1060	85	313,446.6	1	0.103	24	0.044	10	0	10	0	10	0	10	0
1060	90	302,479.1	0	0.161	27	0.021	10	0	10	0	10	0	10	0
1060	95	292,282.6	0	0.187	2	0.045	9	0.000	10	0	7	0.001	10	0
1060	100	282,536.5	0	0.281	1	0.097	9	0.004	7	0.004	7	0.002	9	0.000
1060	105	273,463.3	0	0.325	8	0.047	8	0.000	6	0.001	5	0.001	10	0
1060	110	264,959.6	0	0.291	0	0.044	8	0.000	9	0.000	9	0.000	10	0
1060	115	256,735.7	0	0.310	2	0.063	8	0.001	7	0.001	7	0.002	10	0
1060	120	249,050.5	0	0.327	2	0.075	1	0.008	3	0.005	5	0.004	10	0
1060	125	241,880.4	0	0.297	0	0.078	9	0.000	7	0.002	3	0.008	10	0
1060	130	235,203.4	0	0.317	0	0.108	5	0.006	7	0.003	8	0.002	10	0
1060	135	228,999.2	0	0.339	0	0.101	7	0.001	6	0.001	2	0.006	9	0.001
1060	140	223,062.0	0	0.357	0	0.081	8	0.000	9	0.000	4	0.002	10	0
1060	145	217,462.8	0	0.372	0	0.103	4	0.001	10	0	2	0.007	8	0.000
1060	150	**212,230.5**	0	0.310	0	0.067	6	0.004	4	0.004	1	0.003	10	0
Average:			8.2%	0.1651	43%	0.0457	89%	0.0009	90%	0.0007	87%	0.0008	99%	0.0000

(1) Number of times best known solution found (2) percentage of average solution above best known one.

based heuristic which is the best heuristic algorithm reported in Drezner et al. [22]. Results are also reported for both versions of the VNS algorithm, the genetic algorithm, and the combined

Figure 4: Total Run Times in Minutes

approach, each repeated 10 times. Note that since a random number generator is used different times for each run and each method, the starting solutions are not identical for all methods because the seed for the random number generator is different for each run. For each algorithm the number of times (out of 100 or 10) that the best known solution was found and the percentage of the average solution above the best known solution are reported in Table 3. The total run times in minutes (including the generation of 100 IMP solutions for each run) are reported in Table 4. For

Table 4: Average Run Times (minutes)

Procedure	$n = 654$	$n = 1060$	All
100 IMPs	196.0	589.9	447.4
IDEC3	310.7	1,143.3	842.2
BVNS	202.1	716.3	530.3
DVNS	158.4	564.5	417.6
GA	29.0	182.2	126.8
COMB	143.3	532.2	391.5

example, generating 100 IMP solutions takes an average of 4.47 minutes, and run time for GA is 12.68 minutes per run which means that only an average of 8.21 minutes is spent on Steps 2-9 of the genetic algorithm.

The average percentage above the best known solution dropped from 0.1651% for the starting solutions to 0.0457% by IDEC3 and only 0.0009% for BVNS, 0.0007% for DVNS, 0.008% for GA, and 0.0000% (it is actually 0.000016%) for COMB. COMB found the best known solution in 466 out of 470 runs. Run times for the BVNS, DVNS, and COMB algorithms are shorter than those required for IDEC3. Run times for GA are much shorter. The average run time for the DVNS algorithm was lower by 22.7% than the average for BVNS which is very significant ($p - value = 4 \times 10^{-26}$) while producing slightly better results. The GA gave the best performance when run time is contrasted with the quality of the solutions. COMB gave the best quality results while requiring the second shortest run time.

We also tested GA and COMB with $G = 100p$ rather than $G = \frac{np}{5}$. The GA and COMB results reported in Tables 3 and 4 changed as follows: For GA, the percentage of times the best known solution was found dropped from 87% to 86%, the average above best known solution increased from 0.0008% to 0.0009%. However, average run time decreased by about 30% from 12.68 minutes per run to 8.99 minutes per run. This is actually a reduction of about 45% from 8.21 minutes to 4.52 minutes for the process following the generation of the starting solution. For COMB, the percentage of times the best known solution was found dropped from 99% to 97%, the average above best known solution increased from 0.0000% to 0.0001%. However, average run time decreased by about 10% from 39.15 minutes per run to 35.24 minutes per run.

5 Suggestions for Further Improvements of the Heuristics

The tested instances leave very little room for improvement in solution quality (see the COMB results in Table 3). One avenue for future research is to construct algorithms with similar solution quality but shorter run times. A second avenue is to solve larger problems.

The selection of $k_{\max} = 20$ may be too high because few improvements of the solution in the VNS algorithms occur for $k > 10$ (see also Tables 1 and 2). One option, that will also reduce the number of required parameters, is to estimate the mode of the values of k which result in improved solutions and set $k_{\max}$ as double this value. This entails using $m = 0.5$ in (9). Since the performance is not that sensitive to the selection of λ, when using $\lambda = 2$, the transformation $\psi(x)$ and the density function $\phi(x)$ are:

$$\psi(x) = x\left[x^2 - \frac{3}{2}x + \frac{3}{2}\right] \tag{14}$$

$$\phi(x) = \frac{1}{\pi\left[x^2 - x + \frac{1}{2}\right]} \tag{15}$$

Such a simple approach may be effective for designing DVNS for the solution of other problems without increasing the number of required parameters. Note that the expression for $\phi(x)$ is not required for the implementation of DVNS.

There seems to be a potential for improving the genetic algorithm because the proposed algorithm uses a very basic approach. The success of the proposed genetic algorithm is due, in part, to the simple and effective merging procedure. If an improved merging procedure is constructed, it may improve the GA results.

Some of the following suggestions have been tried but so far we were not able to improve the genetic algorithm without significantly increasing the running time. It is possible that appropriate fine tuning of the parameters may yield good results.

1. The IALT local search is replaced by the more powerful IMP local search.

2. When a new best solution is found, the more powerful IMP local search is applied on the newly found best solution before entering it into the population.

3. A quick VNS algorithm is applied on the merged offspring rather than applying IALT or IMP.

4. The merging process can be viewed as taking about half the facilities from each parent by drawing a line through the "center" of the cluster of the facilities. Most of the selected facilities cover the associated demand points in a good spatial distribution, but there is a "strip" near the dividing line where the facilities may not be distributed properly. The merging procedure may be improved by selecting from each parent, for example, about 40% of the facilities farthest from the dividing line. The remaining 20% of the facilities (about 10% from each parent) are facilities in the "strip". Demand points whose closest facility is a facility in the strip are defined as demand points in the strip. The additional 20% of the facilities which serve demand points in the strip can be generated by some rule.

5. The genetic algorithm performed better for smaller values of p. The merging process selects half of the configuration from each parent and when p is large, it is possible that half of the configuration is too large to capture good parts from each parent. It may be possible to design a merging process that extracts smaller "chunks" of the configuration. Merging more than two parents, thus using smaller fragments from each parent, does not seem appropriate. A scheme can be designed where about a quarter (a parameter) of the facilities are selected from one parent and three quarters from the other. This is similar to the discussion in [27] where the authors suggest that it is beneficial that one parent (the male) is smaller than the other parent (the female) even though they contribute the same number of chromosomes. This can be done in several ways.

   (a) When $p > 100$ select 50 facilities (rather than $\frac{p}{2}$) from the first parent and $p-50$ facilities from the second parent.

   (b) Following the rotation of the system of coordinates the projections on the x-axis and the y-axis are performed and the median value of each projection is calculated. The area is divided into four regions by perpendicular lines through the two medians. Each region contains about one quarter of the facilities. We then can merge one region from one parent and three regions from the other for a total of eight merged solutions. The best

of these eight merged solutions is selected as the offspring. The six pairs of regions, two from each parent (which includes the currently used merging process), can be added to the scheme and the best of the 14 merged solutions selected as the offspring.

(c) The facilities of each parent are merged into a set of $2p$ facilities. The facilities are matched into p pairs, each pair consisting of one facility from each parent. A facility is randomly selected from the combined set of $2p$ facilities. One facility is selected from each pair. About $\frac{1}{4}p$ of the facilities closest to the selected facility are selected from the first parent, and the remaining about $\frac{3}{4}p$ of the facilities farther from the selected facility are selected from the second parent. This is similar to the merging process in [19].

6. Another merging process can follow the idea in Alp et al. [1] that was successfully applied in the genetic algorithm for the solution of the p-median problem in a network environment. They suggested to merge the two sets of facilities to a combined set of $2p$ facilities. The facilities that are located at the same locations in both parents are selected for the offspring. The number of non-selected facilities is reduced by dropping one facility at a time according to some criterion (for details see [1]) until the number of facilities is reduced to p. This idea can be implemented for the planar p-median problem by crafting a similar rule.

We observed that the diversity of the population of the genetic algorithm deteriorates too quickly. The members of the population become too similar to one another. It is possible to apply various established techniques to slow down the population's convergence (for example, [20, 17, 23]).

6 Conclusions

Four heuristic procedures are proposed for solving the planar p-median problem: two versions of variable neighborhood search, including a new "distribution based" approach, a genetic algorithm, and a combination of both. The genetic algorithm was the fastest and solved almost perfectly problems with up to 100 facilities. A problem with 1060 demand points and 100 facilities was solved in 20.95 minutes (when using fewer generations ($100p$), run time fell to 14.36 minutes) and found the best known solution in all 10 runs. Larger problems were solved the best by the hybrid

approach combining the genetic algorithm and the distribution based variable neighborhood search, but required longer run times.

Acknowledgement The authors would like to thank the anonymous referees for their interesting suggestions that improved the content as well as the presentation of the paper.

References

[1] Alp, O., Drezner, Z., and Erkut, E. (2003). An efficient genetic algorithm for the p-median problem. *Annals of Operations Research*, 122:21–42.

[2] Bongartz, I., Calamai, P. H., and Conn, A. R. (1994). A projection method for ℓ_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., Mladenovic, N., and Salhi, S. (2012). Generating good starting solutions for the p-median problem in the plane. *Electronic Notes in Discrete Mathematics*, 39:225–232.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] Brimberg, J. and Mladenović, N. (1999). Degeneracy in the multi-source Weber problem. *Mathematical Programming*, 85:213–220.

[10] Carlsson, S. (1984). Improving worst-case behavior of heaps. *BIT Numerical Mathematics*, 24:14–18.

[11] Carrizosa, E., Dražić, M., Dražić, Z., and Mladenović, N. (2012). Gaussian variable neighborhood search for continuous optimization. *Computers & Operations Research*, 39:2206–2213.

[12] 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.

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

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

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

[16] Das, S. and Suganthan, P. N. (2011). Differential evolution: A survey of the state-of-the-art. *IEEE Transactions on Evolutionary Computation*, 15:4–31.

[17] Drezner, T. and Drezner, Z. (2006). Gender specific genetic algorithms. *INFOR, Information Systems and Operations Research*, 44:117–127.

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

[19] Drezner, Z. (2003). A new genetic algorithm for the quadratic assignment problem. *INFORMS Journal on Computing*, 15:320–330.

[20] Drezner, Z. (2005). A distance based rule for removing population members in genetic algorithms. *4OR - A Quarterly Journal of Operations Research*, 3:109–116.

[21] Drezner, Z. (2013). The fortified Weiszfeld algorithm for solving the Weber problem. *IMA Journal of Management Mathematics*. In press.

[22] Drezner, Z., Brimberg, J., Salhi, S., and Mladenović, N. (2013). New local searches for solving the multi-source Weber problem. in review.

[23] Drezner, Z. and Marcoulides, G. A. (2003). A distance-based selection of parents in genetic algorithms. In Resende, M. G. C. and de Sousa, J. P., editors, *Metaheuristics: Computer Decision-Making*, pages 257–278. Kluwer Academic Publishers.

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

[25] Drezner, Z. and Salhi, S. (2002). Using hybrid metaheuristics for the one-way and two-way network design problem. *Naval Research Logistics*, 49:449–463.

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

[27] Epelman, M. A., Pollock, S., Netter, B., and Low, B. S. (2005). Anisogamy, expenditure of reproductive effort, and the optimality of having two sexes. *Operations research*, 53:560–567.

[28] Gabow, H. N., Galil, Z., Spencer, T., and Tarjan, R. E. (1986). Efficient algorithms for finding minimum spanning trees in undirected and directed graphs. *Combinatorica*, 6:109–122.

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

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

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

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

[33] 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.

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

[35] 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.

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

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

[38] Salhi, S. and Gamal, M. D. H. (2003). A genetic algorithm based approach for the uncapacitated continuous location–allocation problem. *Annals of Operations Research*, 123:203–222.

[39] 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.

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

[41] Xiao, Y. Y., Zhao, Q. H., and Mladenović, N. (2013). Variable neighborhood simulated annealing algorithm for capacitated vehicle routing problems. *Engineering Optimization*. In press.