Journal of Global Optimization manuscript No.
(will be inserted by the editor)

Solving the Planar p-Median Problem by Variable Neighborhood and Concentric Searches

Zvi Drezner[1], Jack Brimberg[2], Nenad Mladenović[3] and Said Salhi[4]

[1] Steven G. Mihaylo College of Business and Economics, California State University-Fullerton, Fullerton, CA 92834.
[2] Department of Mathematics and Computer Science,The Royal Military College of Canada, Kingston, ON Canada.
[3] Mathematical Institute, Serbian Academy of Sciences and Arts, Belgrade, Serbia.
[4] Centre for Logistics & Heuristic Optimization, Kent Business School, University of Kent, Canterbury CT2 7PE, United Kingdom

Received: date / Revised version: date

Abstract Two new approaches for the solution of the p-median problem in the plane are proposed. One is a Variable Neighborhood Search (VNS) and the other one is a concentric search. Both approaches are enhanced by a front-end procedure for finding good starting solutions and a decomposition heuristic acting as a post optimization procedure. Computational results confirm the effectiveness of the proposed algorithms.

Key Words: Location-allocation; Variable Neighborhood Search; Concentric Tabu; Concentric Search; Planar p-median.

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 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.

The single facility version is the oldest known location problem with a history traced all the way back to Fermat in the 1600s (Drezner et al. (2002); Wesolowsky (1993)). In modern times the problem was presented

by Weber (1929) who suggested its application to the location of a service facility, such as a distribution center, serving a set of demand points. The weights represent the number of customers located at each demand point and the objective is to minimize the total distance traveled by all customers. When several facilities are to be built to serve a set of demand points, it is assumed that every customer will seek the service from the closest facility thus defining the multi-source Weber problem.

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:

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

Krau (1997) developed an exact algorithm and optimally solved problems with $n = 50$ and 287 demand points and any number of facilities. Drezner (1984) and Chen et al. (1998) developed optimal solution procedures for the location of $p = 2$ facilities. Schöbel and Scholz (2010) optimally solved problems with $p = 2, 3$ facilities. The continuous p-median problem (1) is known to be NP-hard (Megiddo and Supowit (1984)), and as a result, many heuristics have been developed to solve it. Classical heuristics include the famous alternating procedure by Cooper (1963, 1964), also the projection method of Bongartz et al. (1994), and gradient-based methods such as Murtagh and Niwattisyawong (1982) and Chen (1983).

Brimberg and Drezner (2013) proposed several heuristics for solving the p-median problem. One of the approaches is IALT which is a modification of Cooper's alternate algorithm (Cooper (1963, 1964)). Brimberg et al. (2014) developed the reformulation local search (RLS) which is a new local search that iterates between the continuous problem and a discrete approximation. This is achieved by artificially adding the new found Weber points as potential sites when solving the discrete version of the p-median problem. Drezner et al. (2013) proposed a constructive algorithm START and a decomposition approach. A procedure that combines the three approaches above is termed "Decomposition using START and RLS" (DSR). The details of DSR are given in Drezner et al. (2013). For recent reviews of solution approaches to the continuous p-median problem the reader is referred to Brimberg et al. (2000, 2008, 2014); Drezner et al. (2013).

1.1 Contributions of This Paper

In this paper we propose an improved version of VNS and introduce a "concentric search" for the solution of planar p-median problems. The concentric search, which can be viewed as a variant of VNS, was not applied, to the best of our knowledge, to any optimization problem other than the quadratic

assignment problem (Drezner (2005)) for which it was designed. We anticipate that researchers will find the concentric search useful to solving other optimization problems.

This paper also examines the effect on performance of the quality of the starting solution. We also examine the usefulness of a powerful post-optimization local search.

In summary, this paper develops a new concentric search which has many features of the Variable Neighborhood Search (VNS). We then compare the concentric search to a basic VNS in extensive computational experiments. We also examine the effect on performance of using good starting solutions and of including a powerful post-optimization scheme. The ideas are readily extended to other continuous location problems. In Section 2 we provide short outlines of existing algorithms used in this paper. In Section 3 the tested version of the VNS is described and in Section 4 the concentric search is detailed. We then report computational experiments with these two algorithms and draw some conclusions.

2 An Overview of Our Methodology

Our approach consists of three main steps applied in sequence.

Step 1: Generation of an initial solution using a suitable heuristic. The heuristic used in this paper is START (Drezner et al. (2013)) which is a constructive heuristic with a random component.

Step 2: Improving the starting solution. In this paper we designed (i) a VNS based method, and (ii) a concentric search which is applied to the planar p-median problem for the first time.

Step 3: Improving the solution further using a post-optimization procedure. In this paper we employed the DSR heuristic (Drezner et al. (2013)) which is based on the Reformulation Local Search proposed in Brimberg et al. (2014), and the Delaunay triangulation proposed in Lee and Schachter (1980).

Each of the heuristics incorportaes the local search IALT (Brimberg and Drezner (2013)) which is an improvement on on the locate-allocate ALT heuristic suggested by Cooper (1963, 1964).

In the following subsections we present the way we generate the initial solution (Step 1), the local search used within Step 2, and our powerful post optimizer. The two metaheuristics (VNS and Concentric) are described in the next two sections.

2.1 Generation of Initial Solutions by START (Step 1)

The outline of START is as follows. For complete details the reader is referred to Drezner et al. (2013). 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 $i < j$

$$\Delta_{ij} = \frac{v_i v_j}{v_i + v_j} d_{ij}(1 + u) \tag{2}$$

where u is a random variable in $[0, 1]$ that is used as a stochastic perturbation.

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 (Carlsson (1984); Gabow et al. (1986)) for faster execution. Once the p clusters are formed, each defines a facility and the solution is improved by the IALT algorithm described next.

2.2 The Local Search IALT (A Subroutine in Steps 2 and 3)

The outline of IALT is as follows. For complete details the reader is referred to Brimberg and Drezner (2013).

A parameter L is given. We selected $L = \min\{n, 20\}$. Initial locations for p facilities are generated. Two vectors are maintained throughout the process. An assignment vector of length n indicating which facility is serving demand point i, $i = 1, \ldots, n$. An indicator vector of length p associated with the facilities indicating whether the optimal location of the facility was found (i.e, the subset of demand points associated with the facility did not change since the location of the facility was found). An indicator of zero means that the location of the facility may not be optimal and an indicator of one means that it is optimal. The indicator vector is initially set to all zeroes. The algorithm is outlined in Figure 1.

2.3 The Post-Optimization Procedure DSR (Step 3)

A solution consists of a set of p facilities. Sets of "adjacent" $q = 3$ or 4 facilities are generated. See Figure 2 for an example of $p = 5$ facilities. The set of p facilities is triangulated by the Delaunay triangulation (Lee and Schachter (1980)) obtaining a list of triangles (there are 4 triangles in Figure 2). All pairs of triangles that have a common edge are found (there are 4 such pairs in Figure 2). The union of such a pair of triangles is a quadrangle. If the quadrangle is convex (two such pairs in Figure 2), the two triangles formed by the other diagonal of the quadrangle are added to

Fig. 1 The IALT Algorithm

1. Each demand point is assigned to the closest facility partitioning the set of demand points to subsets each attracted to a facility.
2. A facility with an indicator of zero is randomly selected.
3. The Weiszfeld algorithm (Weiszfeld (1936)) as accelerated in Drezner (1996) and in Brimberg and Drezner (2013) is used to find the optimal location of the selected facility for the subset of demand points currently attracted to that facility. The facility is relocated and the indicator for the relocated facility is set to one.
4. The demand points are re-allocated to their closest facilities. For each demand point that changed an assignment, the two facilities involved in the change (including possibly the facility that just has been relocated) get an indicator of zero. If a zero indicator exists, go to Step 2.
5. For each demand point i the difference between the shortest distance to a facility and the second shortest distance to another facility δ_i is calculated.
6. The L smallest values of δ_i are selected for transfer starting at the smallest δ_i and continuing in order. A transfer consists of transferring the assignment of a demand point from its closest facility to its second closest and finding the new locations for these two facilities based on the new subsets. For each δ_i:
   (a) If a transfer fails to improve the value of the objective function, the transferred point is returned to its original subset.
   (b) If a transfer leads to an improved value of the objective function, it is performed. Reassign all demand points to their closest facility. All facilities whose original subset is unchanged get an indicator of one while those whose subsets have changed get an indicator of zero. Go to Step 2.
7. If all L transfers fail to improve the value of the objective function the algorithm terminates.

the list of triangles, and the quadrangle is added to the list of quadrangles. If the quadrangle is not convex (two such pairs in Figure 2), it is a triangle with a facility in its interior. This set of four facilities is also added to the list of quadrangles. In Figure 2 we obtain 8 triangles and 4 quadrangles. The post optimization procedure is described in Figure 3.

It is proven in Drezner et al. (2013) that the number of polygons in P is bounded by $11p - 32$ ($8p - 23$ triangles and $3p - 9$ quadrangles). Note that by definition $p \geq 3$. Otherwise, there is no triangulation possible. For $p = 3$ there is only one triangle and no quadrangles. The bound in this case is tight. The number of polygons is usually much lower than this bound.

3 The Variable Neighborhood Search

The shaking mechanism in our proposed implementation of the VNS algorithm is based on one of the heuristics given in Brimberg et al. (2000). It

Fig. 2 Generating the Set of Polygons

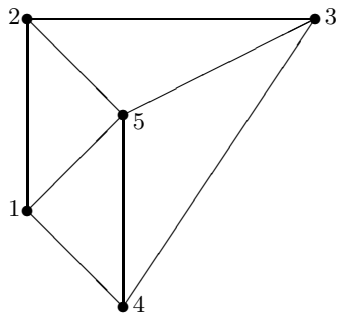

Fig. 3 The DSR Algorithm

1. Generates a list of polygons P.
2. Repeat the following for each of the polygons in P (in random order but selecting triangles first and once all triangles have been examined, selecting quadrangles):
   (a) Find the union U of all demand points served by $q = 3$ or 4 facilities defining the polygon.
   (b) Find an initial solution for q facilities in U by START and apply the Reformulation Local Search (RLS) (Brimberg et al. (2014); Drezner et al. (2013)) getting a heuristic solution to the q-median problem for U.
   (c) If the value of the objective function of the new solution for the q facilities is better than the value of the objective function of the original configuration serving U, update the location of the q facilities, apply IALT on the complete set of p facilities and go to Step 1.
   (d) Otherwise, continue with the next polygon in the list.
3. If the process fails to improve the value of the objective function for all polygons, the algorithm terminates.

was proposed by Brimberg and Mladenović (1995) based on the exchange heuristic proposed by Love and Juel (1982). We improved the algorithm by using more than one solution in the 'shaking' and the 'move or not move' steps. Also we incorporate a more powerful local search.

A starting solution is given and serves as the current solution. For a given solution, the demand points are partitioned into p subsets each closest to one of the facilities.

A perturbation is defined as selecting two demand points in different subsets and exchanging between them.

The k^{th} neighborhood of the current solution is the result of performing a sequence of k perturbations. A demand point may be selected again in subsequent perturbations.

The neighborhoods are tested in order for $1 \leq k \leq k_{\max}$. For each neighborhood, b members are randomly selected. The local search IALT described earlier is applied on each neighborhood member. This process is repeated m times. Once an improved solution is found, it replaces the current solution and the whole process restarts. The count of the number of repetitions starts from 0. The algorithm terminates when the current solution (the original starting solution or a replaced one) is not improved in all m repetitions. The VNS algorithm is described in Figure 4.

Fig. 4 The VNS Algorithm

1. Input an initial solution X_c.
2. Set $k = 1$ and $j = 0$.
3. Repeat the following up to b times:
   (a) Select a solution X at random in the k^{th}-neighborhood of X_c.
   (b) Run IALT on X to obtain a local minimum X_L.
   (c) If X_L is a better solution than X_c, set $X_c = X_L$, and go to Step 2.
4. If none of the b solutions is better than X_c:
   (a) Set $k = k + 1$.
   (b) If $k \leq k_{\max}$, return to Step 3.
   (c) Otherwise, set $j = j + 1$. If $j \leq m$, set $k = 1$ and return to Step 3.
   (d) Else stop (final solution is X_c).

The procedure requires three parameters: $k_{\max}$, b, and m.

4 The Concentric Search

The concentric search, originally termed concentric tabu search by Drezner (2002), was applied in Drezner (2003) and extended in Drezner (2005) for the solution of the quadratic assignment problem (see for example, Drezner et al. (2005)). To the best of our knowledge it was not applied to other optimization problems. Concentric search has many features of the Variable Neighborhood Search (VNS) (Hansen and Mladenović (1997); Mladenović and Hansen (1997)) and can be viewed as a variant of VNS. In VNS shaking of the center solution is defined by randomly selecting a perturbed solution with increasing "distance" from the best known solution, while in concentric search finding randomly a solution in neighborhood $k + 1$, is accomplished by shaking once the best known solution in neighborhood k. Another way to view the concentric search is to compare it to the dynamic programming approach. The optimal solution is in the neighborhood of a certain radius. Suppose we save for each radius k a list of all possible solutions for that neighborhood. The process will then find the optimal solution if the list for radius $k+1$ is generated by all possible perturbations of solutions in the list of radius k and the process continues for all possible radii. In the concentric

search we "assume" that a good solution will be generated by restricting the process of generating the list of radius $k+1$ by perturbations of only the best found solution in the list of radius k.

We present this variant to the VNS community so it can be applied for the solution of other optimization problems.

4.1 Introduction to the Concentric Search

The starting solution is termed the "center" solution and is the best found solution. In a particular solution, each demand point is closest to a facility. Therefore, each facility defines a subset of demand points closest to it. A "radius" around the center solution is defined similarly to the sequence of neighborhoods in VNS. The radius is the number of demand points not in their center solution subset.

In the original concentric tabu search (Drezner (2002)) which was applied for the solution of the quadratic assignment problem, the once perturbed (basic) neighborhood consists of all pairwise exchanges between two facilities. Therefore, solutions in the basic neighborhood of a solution of radius k can have a radius of $k-2$, $k-1$, k, $k+1$, $k+2$. Rather than selecting a solution in a neighborhood of radius k randomly, the best solution with radius $k+1$ is found in the basic neighborhood of the best solution of radius k (or $k-1$).

During the search of the neighborhood of radius k, solutions with a radius lower or equal to k are "tabu" meaning that they are ignored unless they are better than the best found (center) solution. Observe that neighborhoods for a large radius are in different regions of the solution space. As in VNS, constructed solutions move "farther" and "farther" away from the center solution. If a solution better than the current center solution is found, it becomes the new center solution and the process is restarted. The whole process can be repeated K (a parameter) times with the best solution for $k = k_{\max}$ used as the new center solution. In the present paper we just experimented with $K = 1$.

4.2 Comparing Variable Neighborhood Search to Concentric Search

In VNS the perturbations are performed on the center (best found) solution while in concentric search the perturbations are performed on perturbed solutions. In order to search far away in the solution space from the best found solution, a sequence of many perturbations is required. In concentric search the perturbations are performed on perturbed solutions that may be already quite far from the center solution, thus a relatively small number of perturbations is required to move the search far away from the center solution.

4.3 Implementation of the Concentric Search

We modify the original concentric tabu search to adjust it to the special structure of the location-allocation problem. Since the number of solutions in the basic neighborhood can be very large, we propose to randomly select m (a parameter similar to b in VNS) solutions in the basic neighborhood rather than all of them. Another difficulty is that when a solution in the neighborhood is selected, a local search such as IALT (Brimberg and Drezner (2013)) is applied to it, and the resulting solution may have a different radius. Yet another issue is that when IALT is applied on the perturbed solution and we use only $k = 1$ in the basic neighborhood, IALT may return to the center solution especially if it is a good one. We therefore define the neighborhood as exchanging t (a parameter) pairs of demand points between subsets rather than only one. These t pairs are selected randomly and the same demand point may be selected again. This is identical to the t^{th} neighborhood in VNS. However, the perturbation (shaking) is performed on the best found solution in the k^{th} neighborhood rather than on the center solution.

There are three variants of the concentric search:

(i) the original one (Drezner (2002)), termed "CON(Orig)", moves the search only to greater values of k;
(ii) the ring moves (CON(RM), Drezner (2005)) allows the repetition of the search for the same k if the best known solution for that k was improved while scanning all members of the basic neighborhood of k;
(iii) all moves (CON(AM), Drezner (2005)) which allows the search to proceed with a lower value of k if the best known solution for that k was improved.

In all variants, a list of the best solutions for each radius $k \leq k_{\max}$ is stored. An indicator is maintained for each list member that indicates whether its basic neighborhood was already evaluated (indicator of 1) or not (indicator of 0). If a solution better than the best found solution (which is the center solution) is found during the process, the center solution is replaced and the whole process restarts.

4.4 The Concentric Algorithm

The parameters $k_{\max}$, the number m of randomly selected members in the neighborhood and t, the neighborhood radius, are given.
Let

k be the radius of a solution (the number of demand points not associated with same facility as the center solution). The starting (center) solution is defined with $k = 0$.

I_k be an index vector for $k = 0, \ldots, k_{\max}$. $I_k = 1$ if the neighborhood of the best known solution with radius k has already been evaluated, and $I_k = 0$ otherwise.

f_k be a vector for of the values of the objective function of the best known solution with radius k, $k = 0, \ldots, k_{\max}$.

$list_k$ be a list of length n of the subset assigned to each demand point in the best known solution with radius k for $k = 0, \ldots, k_{\max}$.

The concentric Algorithm is described in Figure 5.

Fig. 5 The Concentric Algorithm

1. Set: $I_0 = 0$ and $I_k = 1$ for $k = 1, \ldots, k_{\max}$; f_0 to the value of the objective function for the center solution; $list_0$ to the list of the center solution.
2. Select the smallest $0 \le k \le k_{\max}$ for which $I_k = 0$.
3. If no such k exists, stop with the center solution as the output of the algorithm.
4. Otherwise, set $i = 1$:
   (a) Create a list $list^* = list_k$ of the subset assignments of the best found solution with radius k.
   (b) (shaking operation) Repeat the following t times:
      - Randomly select a pair of demand points with different assigned subsets in $list^*$,
      - Perturb the solution by exchanging between them and update $list^*$.
   (c) Apply IALT on the perturbed solution yielding a solution with an objective function f.
   (d) If $f < f_0$, replace the center solution and go to Step 1.
   (e) Find the radius r of the resulting solution by comparing its list with $list_0$.
   (f) For CON(Orig): if $r \le k$ or $r > k_{\max}$ go to Step 4h.
   For CON(RM): If $r < k$ or $r > k_{\max}$ go to Step 4h.
   For CON(AM): If $r > k_{\max}$ go to Step 4h.
   (g) If $f < f_r$, then update $f_r = f$, $list_r$ and set $I_r = 0$; if $r = k$ go to Step 2.
   (h) Set $i = i + 1$ and if $i \le m$ go to Step 4a.
5. Set $I_k = 1$ and go to Step 2.

5 Computational Experiments

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

which is required for DSR, was coded using a Fortran program based on Sugihara and Iri (1994) using subroutines first developed by Ohya et al. (1984). We thank Atsuo Suzuki for providing and modifying the triangulation Fortran program.

We tested the algorithms on the four problems given in Brimberg et al. (2000). The first one ($n = 50$) is the well-studied 50-customer problem from Eilon et al. (1971). The second one ($n = 287$) is the ambulance problem from Bongartz et al. (1994); the last two ($n = 654$ and 1060) are from the TSP library (Reinelt (1991)). The best known solutions are taken from Brimberg et al. (2000); Taillard (2003); Brimberg et al. (2014). These solutions were proven optimal by Krau (1997) for the $n = 50$ and 287 instances investigated. Our computational results for $n = 50$ were all optimal, and thus, not reported here. We tested 32 values of p for the $n = 287$ problem listed in Table 2, 28 values of p for the $n = 654$ problem listed in Table 3, and 20 values of p for the $n = 1060$ problem listed in Table 4, for a total of 80 instances. Each instance was run 10 times from different randomly generated starting solutions.

5.1 Experiments with VNS

We tested the VNS heuristic starting from one START solution (VNS(1)) or from the best of 100 START solutions (VNS(100)). Note that each START solution includes an IALT application at the end. These two variants are designed to evaluate the effect of a "good" starting solution on the performance of the VNS algorithm. We used $m = 100$, 200, and 300. The results are summarized in Table 1. We applied $k_{\max} = \min\{p, 20\}$, $b = 10$, and tested various values of the stopping condition m. Note that in the standard VNS, $b = 1$ is used and thus no such parameter is defined.

5.2 Experiments with the Concentric Algorithm

We tested the three variants of the concentric search (CON(Orig), CON(RM), and CON(AM)). The starting solution used the best of 100 runs of START. We also tested a starting solution obtained by one application of START, but the results were significantly inferior and thus not reported. Each variant was repeated ten times for each value of p selected for each problem. The results are also summarized in Table 1.

The selection of t is quite tricky. If t is too small, the subsets are only slightly modified and IALT may terminate with the center solution. If t is too large, the perturbed solution may resemble a random solution that has lost some of the structure of the center solution. Following extensive experiments we recommend $k_{\max} = n/2$, $t = 2n/p$ for perturbing the center solution, and $t = 2 + \frac{n}{100}$ (i.e., $t = 4, 8, 12$ for $n = 287, 654, 1060$, respectively) for perturbing other solutions. The quality of the solution and run time depend on the value of m. We tested $m = 100, 200, 300$.

5.3 Results

The average for all values of p for each n are reported in Table 1: (i) percentage of the minimum result and the average result above the best known solution and (ii) average run time per one run per instance. Percentages are reported to three decimal digits. If the value is "exactly" zero we report it as 0, and if it rounds to zero to three decimal places we report it as 0.000%. In Tables 2-4 we report results for specific variants of the algorithm for each set of problems.

5.4 Discussion of Results

Comparing the VNS trials that begin with one run of START to those with the best of 100 runs of START clearly shows that selecting a better starting solution significantly improves the final VNS result. Run times for START are very short (Drezner et al. (2013)): 100 runs of START require about 0.03, 0.7, 3, and 13 seconds for the $n = 50, 287, 654, 1060$ problems, respectively. Furthermore, the total execution time for VNS starting from the best of 100 runs of START is *shorter* in most cases because the starting solution is better and fewer iterations of VNS are performed.

Both VNS and the concentric algorithms performed very well solving the $n = 287$ instances. The optimal solution was found for each of the 32 instances by both VNS(100) and the concentric procedures. The concentric algorithm performed better than VNS both in the quality of the solutions and shorter run times. Both provide much better results than those reported in published papers. In Table 2 we report the optimal solution and run time in seconds per run for the $n = 287$ instances using CON(AM) and $m = 200$. 317 of the 320 runs found the optimal solution. The three cases for which the optimal solution was not found are: (i) for $p = 75$, 9 out of 10 runs were optimal and the average was 0.0006% above optimum, and (ii) for $p = 85$, 8 out of 10 results are optimal and the average is 0.011% above optimum. The average for all 32 values of p was therefore 0.00036% above optimum. The results did not change by applying DSR. We therefore report only the run times (including DSR) in seconds per run for each instance. Performing all 320 runs reported in Table 2 required less than 200 minutes of computing time.

VNS performed slightly better on the $n = 654$ instances. The best average of the best found solution and the average solution above the best known solution was best for VNS(100)+DSR using $m = 300$. Run times were also generally lower for the VNS(100)+DSR algorithm. The detailed results for this variant are reported in Table 3.

Results for the $n = 1060$ instances show about the same performance with a slight edge to the concentric search. Also, the run times for the concentric search are lower than those for VNS. The best averages above the best known solution were obtained for CON(AM)+DSR using $m = 300$. The details of these results are reported in Table 4.

Table 1 Average Performance of 10 Runs of the Algorithms

Method	$n = 287$			$n = 654$			$n = 1060$		
	Min†	Ave†	‡	Min†	Ave†	‡	Min†	Ave†	‡
$m = 100$									
VNS(1)	0.006	0.074	35.3	0.223	1.010	78.9	0.411	1.111	145.4
VNS(1)+DSR	0.006	0.022	35.5	0.125	0.512	79.5	0.251	0.683	148.4
VNS(100)	0	0.012	35.2	0.084	0.250	73.5	0.116	0.284	133.9
VNS(100)+DSR	0	0.010	35.5	0.066	0.189	74.0	0.094	0.240	136.0
CON(Orig)	0	0.007	17.7	0.114	0.295	59.5	0.109	0.279	31.2
CON(Orig)+DSR	0	0.005	18.0	0.053	0.206	60.0	0.058	0.214	33.4
CON(RM)	0	0.008	18.1	0.171	0.320	62.1	0.123	0.278	32.6
CON(RM)+DSR	0	0.007	18.4	0.104	0.215	62.6	0.059	0.221	34.8
CON(AM)	0	0.004	20.5	0.108	0.280	116.7	0.090	0.261	56.5
CON(AM)+DSR	0	0.003	20.7	0.065	0.194	117.2	0.064	0.217	58.6
$m = 200$									
VNS(1)	0.009	0.090	65.2	0.221	0.998	158.0	0.367	1.070	305.0
VNS(1)+DSR	0.009	0.019	65.4	0.118	0.504	158.6	0.158	0.682	308.1
VNS(100)	0	0.005	67.3	0.071	0.210	138.4	0.079	0.257	276.0
VNS(100)+DSR	0	0.005	67.5	0.051	0.166	138.9	0.054	0.215	278.1
CON(Orig)	0	0.002	33.3	0.119	0.277	121.9	0.108	0.274	60.9
CON(Orig)+DSR	0	0.002	33.6	0.075	0.197	122.4	0.076	0.224	63.0
CON(RM)	0	0.001	33.4	0.084	0.277	126.4	0.113	0.273	60.5
CON(RM)+DSR	0	0.001	33.6	0.054	0.211	126.9	0.060	0.226	62.5
CON(AM)	0	0.000	36.8	0.093	0.247	269.9	0.089	0.273	170.6
CON(AM)+DSR	0	0.000	37.0	0.056	0.184	270.4	0.055	0.226	172.7
$m = 300$									
VNS(1)	0.006	0.049	96.0	0.250	0.999	230.6	0.331	1.014	483.2
VNS(1)+DSR	0.006	0.018	96.2	0.130	0.568	231.2	0.186	0.668	486.1
VNS(100)	0	0.006	96.5	0.055	0.207	204.4	0.089	0.250	401.0
VNS(100)+DSR	0	0.005	96.7	0.037	0.160	204.9	0.051	0.206	403.1
CON(Orig)	0	0.001	47.9	0.115	0.254	181.8	0.091	0.254	97.6
CON(Orig)+DSR	0	0.001	48.1	0.047	0.183	182.3	0.064	0.222	99.7
CON(RM)	0	0.002	48.1	0.116	0.272	199.1	0.098	0.263	105.5
CON(RM)+DSR	0	0.002	48.3	0.056	0.196	199.6	0.062	0.215	107.6
CON(AM)	0	0.001	51.7	0.076	0.225	430.4	0.077	0.241	282.7
CON(AM)+DSR	0	0.001	51.9	0.055	0.163	430.9	0.047	0.201	284.7

†Percentage above the best known solution (optimal for $n = 287$)
‡Time (seconds) per run

As expected the post-optimization procedure gave net improvements on the final solution. Although the improvements are relatively small we still recommend this step since the increase in computational times is insignificant. The biggest improvement with the DSR post-optimization procedure is obtained with $m = 100$. Also, computation times are lower with $m = 100$ + DSR than when higher values of m (200, 300) are used without DSR, while the quality of solutions remains comparable.

Table 2 Results for 10 Runs of the $n = 287$ Instances by CON(AM)+DSR with $m = 200$

p	Optimum	†	p	Optimum	†	p	Optimum	†	p	Optimum	†
5	9715.6275	14.0	13	5725.1853	24.1	25	3348.7101	29.2	65	924.5547	51.2
6	8787.5568	14.2	14	5469.6478	26.1	30	2716.9038*	28.9	70	814.2238	54.1
7	8160.3203*	18.5	15	5224.7028	24.9	35	2238.1839	34.9	75	730.0354*	65.7
8	7564.2949	21.7	16	4981.9608	27.1	40	1900.8361	36.2	80	655.3788	50.4
9	7088.1283	22.7	17	4755.1890	25.4	45	1630.3115	39.5	85	588.3680	66.3
10	6705.0356	26.2	18	4547.3651	29.0	50	1402.5836	48.0	90	529.2126	75.7
11	6351.5910	22.8	19	4342.0648	27.4	55	1203.9849	55.3	95	480.8592	68.7
12	6033.0474	26.3	20	4148.8443	26.8	60	1055.1389*	40.0	100	441.2417	63.5

† Time in seconds per run.
* Slightly improved optimal solutions compared with Brimberg et al. (2000).

Table 3 $n = 654$ Results Using VNS(100)+DSR with $m = 300$

p	BK	(1)	(2)	(3)	(4)	p	BK	(1)	(2)	(3)	(4)
5	209068.7935	0	10	0	48.7	35	39257.2685	0	1	0.108	125.4
6	180488.2126	0	10	0	35.4	40	35704.4076	0	1	0.047	139.4
7	163704.1681	0	10	0	38.1	45	32306.9721	0	3	0.184	172.2
8	147050.7904	0	10	0	41.8	50	29338.0106	0	1	0.463	217.7
9	130936.1241	0	10	0	46.7	55	26699.1699	0	1	0.321	194.8
10	115339.0328	0	10	0	50.6	60	24504.3952	0	1	0.223	284.3
11	100133.2007	0	10	0	49.4	65	22747.0996	0.059	2	0.463	324.5
12	94152.0549	0	10	0	55.1	70	21465.4361	0	1	0.226	342.9
13	89454.7613	0	10	0	54.2	75	20312.9668	-0.212	3	-0.080	345.2
14	84807.6690	0	9	0.002	60.0	80	19193.8610	0	1	0.524	521.6
15	80177.0422	0	8	0.005	68.4	85	18316.5391	0.306	1	0.668	512.2
20	63389.0238	0	10	0	103.1	90	17514.4230	0.400	1	0.594	598.3
25	52209.5106	0	10	0	108.1	95	16786.3887	0.173	2	0.381	603.3
30	44705.1920	0	9	0.000	108.5	100	16083.5345	0.065	1	0.346	486.5

(1) Percent of best found solution above best known solution (BK)
(2) Number of times that the best found solution (not BK) was obtained
(3) Percent of average solution above best known solution (BK)
(4) Time in seconds per run

6 Conclusions

Two heuristic algorithms are proposed for solving the planar p-median problem. One is a VNS algorithm and the other is a concentric search which is a variation of VNS. We also add a front-end subroutine for finding "good" starting solutions and a post-optimization procedure that solves decomposed problems based on Delaunay triangulation with a powerful local search. Computational experiments demonstrate the effectiveness of both algorithms. The two algorithms perform consistently well on a wide range of problem instances tested with a small edge to the concentric search. Com-

Table 4 $n = 1060$ Results Using CON(AM)+DSR with $m = 300$

p	BK	(1)	(2)	(3)	(4)	p	BK	(1)	(2)	(3)	(4)
5	1851877.3	0	10	0	35.3	55	422638.7	0.046	1	0.341	63.5
10	1249564.8	0	10	0	25.8	60	397674.5	0.178	1	0.346	78.8
15	980131.7	0	10	0	64.0	65	376630.3	0	1	0.187	76.6
20	828685.7	0	2	0.007	60.6	70	357335.1	0	1	0.171	123.4
25	721988.2	0	3	0.050	58.3	75	340123.5	0	1	0.241	99.6
30	638212.3	0	6	0.059	30.4	80	325971.3	0.001	1	0.165	209.4
35	577496.7	0	6	0.114	22.5	85	313446.6	0.222	1	0.374	291.1
40	529660.1	0	1	0.289	49.3	90	302479.1	0.140	1	0.359	538.2
45	489483.8	0.006	2	0.139	90.0	95	292282.6	0.035	1	0.381	1,394.5
50	453109.6	0.119	1	0.310	54.6	100	282536.4	0.194	1	0.480	2,327.6

(1) Percent of best found solution above best known solution (BK)
(2) Number of times that the best found solution (not BK) was obtained
(3) Percent of average solution above best known solution (BK)
(4) Time in seconds per run

putational results also demonstrate that better end results are obtained on average when better starting solutions are used.

The post-optimization procedure gave a small net improvement on the overall performance of the heuristics. However, since the extra computational effort was relatively insignificant, we recommend that such a procedure be employed.

Results for the $n = 287$ instances are significantly better than those reported in the literature. Results for the $n = 654, 1060$ instances are similar or slightly better than the results reported in the literature.

Acknowledgements We would like to thank the referees for their time and constructive comments that helped to improve the presentation as well as the content of the paper.

References

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.

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

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

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.

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.

Brimberg, J. and Mladenović, N. (1995). A variable neighbourhood algorithm for solving the continuous location-allocation problem. *Cahiers du GERAD*.

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

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.

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

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

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

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

Drezner, Z. (1996). A note on accelerating the Weiszfeld procedure. *Location Science*, 3:275–279.

Drezner, Z. (2002). A new heuristic for the quadratic assignment problem. *Journal of Applied Mathematics and Decision Sciences*, 6:163–173.

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

Drezner, Z. (2005). The extended concentric tabu for the quadratic assignment problem. *European Journal of Operational Research*, 160:416–422.

Drezner, Z., Brimberg, J., Salhi, S., and Mladenović, N. (2013). Effective heuristics for solving the multi-source Weber problem. in review.

Drezner, Z., Hahn, P. M., and Taillard, E. D. (2005). Recent advances for the quadratic assignment problem with special emphasis on instances that are difficult for meta-heuristic methods. *Annals of Operations Research*, 139:65–94.

Drezner, Z., Klamroth, K., Schöbel, A., and Wesolowsky, G. O. (2002). The Weber problem. In Drezner, Z. and Hamacher, H. W., editors, *Facility Location: Applications and Theory*, pages 1–36. Springer, Berlin.

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

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.

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

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

Lee, D. T. and Schachter, B. J. (1980). Two algorithms for constructing a Delaunay triangulation. *International Journal of Parallel Programming*, 9(3):219–242.

Love, R. F. and Juel, H. (1982). Properties and solution methods for large location-allocation problems. *Journal of the Operational Research Society*, pages 443–452.

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

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

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.

Ohya, T., Iri, M., and Murota, K. (1984). Improvements of the incremental method of the Voronoi diagram with computational comparison of various algorithms. *Journal of the Operations Research Society of Japan*, 27:306–337.

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

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.

Sugihara, K. and Iri, M. (1994). A robust topology-oriented incremental algorithm for Voronoi diagram. *International Journal of Computational Geometry and Applications*, 4:179–228.

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

Weber, A. (1929). *ÜBer Den Standort Der Industrien, 1. Teil: Reine Theorie Des Standortes. English Translation: on the Location of Industries*. University of Chicago Press, Chicago, IL. Originally published in Tübingen, Germany in 1909.

Weiszfeld, E. (1936). Sur le point pour lequel la somme des distances de n points donnes est minimum. *Tohoku Mathematical Journal*, 43:355–386.

Wesolowsky, G. O. (1993). The Weber problem: History and perspectives. *Location Science*, 1:5–23.