A local search heuristic for the (r|p)-centroid problem in the plane

I. Davydov a,1, Y. Kochetov a,*, 1, E. Carrizosa b

a Sobolev Institute of Mathematics, Novosibirsk, Russia
b Facultad de Matemáticas, Universidad de Sevilla, Sevilla, Spain

ARTICLE INFO

Keywords:
Competitive facility location
Leader–follower
Bilevel optimization
Stackelberg game

ABSTRACT

In the (r|p)-centroid problem, two players, called a leader and a follower, open facilities to service clients. Clients are identified with their location on the Euclidean plane. Facilities can be opened anywhere in the plane. At first, the leader opens p facilities. Later on, the follower opens r facilities. Each client patronizes the closest facility. Each player maximizes own market share. The goal is to find p facilities for the leader to maximize his market share. It is known that this problem is \( \Sigma_p^2 \)-hard. We develop a local search heuristic for this problem, based on the VNS framework. We apply the \( (r|p)+1 \)-centroid subproblem for finding the best neighboring solution according to the swap neighborhood. It is shown that this subproblem is polynomially solvable for fixed r. Computational experiments for the randomly generated test instances confirm the value of the approach.

© 2013 Elsevier Ltd. All rights reserved.

1. Introduction

This paper addresses a Stackelberg facility location game on Euclidean plane. We assume that the clients demands are concentrated at a finite number of points in the plane and the facilities can be opened anywhere in the plane. In the first stage of the game, a player, called the leader, opens p facilities. At the second stage, another player, called the follower, opens r facilities. At the third stage, each client chooses the closest opened facility as a supplier. In case of ties, the leader’s facility is preferred. Each player tries to maximize own market share but the leader knows how many facilities the follower will locate. The goal of the game is to find p points for the leader facilities to maximize his market share.

The field of competitive location constitutes a broad spectrum of mathematical models, methods, and applications in operations research, economics, and computer science. It is an interesting topic for theoretical studies, experimental research and real-world applications. It is rooted in the work of Hotelling [18] who studied the strategies of two players competing for clients on a line market. For a survey of various competitive facility location models see [12,19].

Such facility location games on a network were studied by Hakimi [15]. Following Hakimi, the leader problem is called a centroid problem and the follower problem is called a medianoid problem. In the literature this game can be found under such names as pre-emptive capture problem [23], competitive location with foresight [21], leader–follower location problem [9], and competitive p-median problem [2], see also [34].

Three types of possible facility locations can be considered:

- at the nodes of a graph (discrete case);
- at the nodes and anywhere on the edges of a graph (absolute case);
- anywhere on a plane (continuous case).

Computational complexity of the game on general graphs is studied in [20,24]. It is shown that the game is \( \Sigma_p^2 \)-hard for the discrete and absolute cases. In [11] these results are strengthened and complemented. Specifically, it is shown that the game is \( \Sigma_p^2 \)-hard for the planar graphs in discrete and absolute cases and in continuous case as well. The follower problem in these three cases is NP-hard in the strong sense. The class \( \Sigma_p^2 \) is a part of the polynomial time hierarchy. It contains all decision problems solvable in polynomial time on a nondeterministic Turing machine with access to an oracle for NP. In particular, this class contains decision problems which can be described using a formula of the form \( \exists x \forall y g(x, y) \), where \( g(x, y) \) is a quantifier-free formula. It is widely assumed that the class \( \Sigma_p^2 \) is a proper superset of the class NP. Thus, the problems from this class turn out to be even more hard than the well-known NP-complete problems [22].

In [7] the alternating heuristic for the centroid problem in the plane is suggested. In each iteration of the heuristic, a solution of one player is considered and the best answer for another player is founded. Two greedy strategies are used to this end. In [10] an improved alternating heuristic is developed. The branch and bound method is applied at each iteration in order to calculate exact market share of the players. At the final stage of the alternating
process, the clients are clustered, and an exact polynomial-time algorithm for the \((r|p|1)\)-centroid problem is applied [13].

In this paper we present a local search heuristic using an exact approach for the follower problem. We consider the \((r|X_{p-1}+1)\)-centroid problem where the leader has \(p-1\) facilities and tries to open an additional facility in the best position. We use this problem in order to find the best neighboring solution in the swap neighborhood. It is shown that this problem is polynomially solvable for fixed \(r\). We solve the medianoid problem for many leader solutions, but the number of such solutions is polynomially bounded. Computational results for randomly generated instances from the benchmark library \textit{Discrete Location Problems} (http://math.nsc.ru/AP/benchmarks/index.html) show that the new approach dominates the previous heuristics. Preliminary version of the paper is presented in the conference proceedings of the EURO MINI Conference XXVII on Variable Neighborhood Search [8].

The paper is organized as follows. Section 2 introduces the relevant notations and states the problem. Section 3 addresses the follower problem and reformulates it as a linear integer programming problem. Section 4 goes on to provide the main theoretical result of the \((r|X_{p-1}+1)\)-centroid problem. Sections 5 and 6 develop the VNS metaheuristic and show the computational results, respectively. Finally, our conclusions are presented in Section 7.

2. Mathematical model

Let us consider a two-dimensional Euclidean plane in which \(n\) clients are located. We assume that each client \(j\) has a positive demand \(w_j\). Let \(X\) be the set of \(p\) points where the leader opens his own facilities and let \(Y\) be the set of \(r\) points where the follower opens his own facilities. The distances from client \(j\) to the closest facility of the leader and the closest facility of the follower are denoted as \(d(j,X)\) and \(d(j,Y)\), respectively. Client \(j\) prefers \(Y\) over \(X\) if \(d(j,Y) < d(j,X)\) and prefers \(X\) over \(Y\) otherwise. By

\[
U(Y<X) = \sum \{ w_j \mid j \in U(Y<X) \}
\]

we denote the set of clients preferring \(Y\) over \(X\). The total demand captured by the follower is given by

\[
W(Y<X) = \sum (w_j)_{j \in U(Y<X)}
\]

For \(X\) given, the follower tries to maximize his own market share. The maximal value \(W^*(X)\) is defined to be

\[
W^*(X) = \max_{Y(X)} W(Y<X).
\]

This maximization problem will be called the follower problem. The leader tries to minimize the market share of the follower. This minimal value \(W^*(X)\) is defined to be

\[
W^*(X) = \min_{X,Y} W^*(X).
\]

For the best solution \(X^*\) of the leader, his market share is \(\sum_{j=1}^{p} w_j - W^*(X^*)\). In the \((r|p)\)-centroid problem, the goal is to maximize the leader market share and find \(X^*\) and \(W^*(X^*)\).

3. The follower problem

Let us first describe an exact approach for the follower problem. Such problem is rewritten as an integer linear programming problem, and solved using the branch and bound method.

For each client \(j\), we introduce a disk \(D_j\) with radius \(d(j,X_j)\) and center in the point where this client is located. Let us consider the resulting intersection of two or more such disks. These disks and their intersections are called regions. The total number of regions is large, but we can eliminate some of them, and consider the convex regions as those defined by intersections only. Thus, we have \(m\) regions and \(m < n(n-1)/2\) [14]. In fact, there are at most \(n(n-1)/2\) pairs of circles with nonempty intersections. Hence, we have at most \(n(n-1)\) intersection points. Each intersection point is adjacent to four regions, and only one of them is convex. Thus, the number of vertices for convex regions is bounded by \(n(n-1)/2\) each region has at least two vertices, hence, we have at most \(n(n-1)/2\) convex regions.

Now we define a binary matrix \((a_{kj})\) to indicate the clients which will patronize a facility of the follower if it is opened inside of a region. Formally, define \(a_{kj} = 1\) if a facility of the follower in region \(k\) captures the client \(j\) and \(a_{kj} = 0\) otherwise. In order to present the follower problem as an integer linear program we introduce two sets of the decision variables:

\[
y_k = \begin{cases} 1 & \text{if the follower opens facility inside of region } k, \\ 0 & \text{otherwise}, \end{cases}
\]

\[
z_j = \begin{cases} 1 & \text{if the follower captures client } j, \\ 0 & \text{otherwise}. \end{cases}
\]

Now the follower problem can be written as the maximum capture problem

\[
\max \sum_{j=1}^{n} w_j z_j,
\]

subject to

\[
z_j \leq \sum_{k=1}^{m} a_{kj} y_k, \quad j = 1, \ldots, n,
\]

\[
\sum_{k=1}^{m} y_k = r,
\]

\[
y_k, z_j \in \{0,1\}, \quad k = 1, \ldots, m, \quad j = 1, \ldots, n.
\]

The objective function gives the market share of the follower. The first constraint guarantees that client \(j\) will patronize a facility of the leader only if the follower has no facility at the distance less than \(d(j,X_j)\). The second constraint allows the follower to open exactly \(r\) facilities. Note that all points inside a region are equivalent for the follower. Thus, we choose one of them as exact coordinates for the follower facility, for example, a middle point for the two vertices from the boundary of the region.

In our computational experiments we observe that the integrality gap is small for this problem in the case of Euclidean plane. The branch and bound method easily finds an optimal solution. For this reason, the exact value \(W^*(X)\) is used in our heuristic for the centroid problem. Note that the follower problem is polynomially solvable for fixed \(r\).

4. Subproblem for a facility of the leader

Let us consider the \((r|X_{p-1}+1)\)-centroid subproblem where the leader has a set of \(p-1\) facilities and wants to open another facility in the best position. We claim that there is a relatively small number of points in the Euclidean plane which we should check for finding this best position. We solve the follower problem for each the point and choose one with the maximal leader market share.

As we have mentioned above, for each client \(j\) we have the disk \(D_j\) with radius \(R_j = d(j,X_{p-1})\). Hence, the plane is divided into regions. When the leader opens a new facility, some of the disks and corresponding regions are modified. More specifically, some of the regions become smaller. Note that all points inside of each region are equivalent for the follower. Thus, we will get a new instance of the follower problem if and only if some of the regions are vanished. Hence, we should check all points for the new
facility where at least one region is vanished. Further, we need the following well-known result for the Euclidean spaces [6].

**Theorem 1 (Helly).** Suppose that \( G_1, \ldots, G_k \) is a finite collection of convex sets of \( d \)-dimensional Euclidean space and \( k > d \). If the intersection of every \( d + 1 \) of these sets is nonempty, then the whole collection has a nonempty intersection.

For \( d = 2 \) this means that intersections of triples of disks determine all intersection of regions. The collection of regions changes if and only if intersections within at least one triple of disks changes.

**Theorem 2.** The \((rX_{p-1} + 1)\)-centroid problem is polynomially solvable for fixed \( r \).

**Proof.** Let \( D_1, D_2, D_1 \) be a triple of disks with radii \( R_1, R_2, R_3 \) and centers in \( j_1, j_2, j_3 \), respectively. Consider all possible cases when at least one region is vanished.

**Case 0:** The disks have no mutual intersections. If the leader opens new facility in one of the points \( j_1, j_2, \) or \( j_3 \), then the corresponding disk is vanished. Hence, we get a new instance of the follower problem. Other points are equivalent and unimportant for the leader.

**Case 1:** Two disks have a mutual intersection, for example, \( D_1 \cap D_2 \neq \varnothing \), but disk \( D_3 \) has no intersections with \( D_1 \) and \( D_2 \). The region \( D_1 \cap D_2 \) is vanished if and only if new facility is opened in interval \( j_1, j_2 \), or in a disk \( D' _1 \) with radius \( R' _1 = \max(0, d(j_1, j_2) - R_2) \), or in a disk \( D' _2 \) with radius \( R' _2 = \max(0, d(j_1, j_2) - R_1) \) if the disks exist (see Fig. 1).

**Case 2:** Each pair of disks has a mutual intersection, but the triple of disks has no intersection. This case is similar to the previous one but now we have to consider two auxiliary concentric disks \( D_j^1, D_j^1, D_j^1 \subseteq D_j^1 \) for each point \( j \) (see Fig. 2). Disk \( D_j^1 \) shows the points for deleting mutual intersections of the disk \( D_j \) with other disks from the triple. The region \( D_j^1 \cap D_j^1 \) saves the intersection with nearest from the disks but deletes the intersection with the other one. As in the previous case, the intervals \( j_1, j_2, j_k \), \( k = 1, 2, 3 \) allow the leader to delete the mutual intersections. Note that the region \( D_j^1 \) is closed while the region \( D_j^1 \cap D_j^1 \) is not closed. It contains its external boundary but does not contain the internal one. As we will see later, this circumstance is not important for our analysis and we will ignore the region’s boundaries.

**Case 3:** The triple of disks has a nonempty intersection. In the previous cases we have got the regions which allow to delete the pairwise intersections and, hence, the intersection of all disks. Now we wish to get the regions for deleting of triple intersection but saving some pairwise intersections. This case can be divided into three disjoint subcases.

**Case 3a:** New facility of the leader is opened in such a way that all three disks are decreased. In other words, we consider the region \( D_1 \cap D_2 \cap D_3 \) (see Fig. 3). Let us denote by \( T \) the triple \( j_1, j_2, j_3 \). Note that each point of the triangle allows to exclude the triple intersection, but save the pairwise intersections. Thus, each point of region \( T \cap D_1 \cap D_2 \cap D_3 \) has required property.

**Case 3b:** New facility of the leader is opened in such a way that only two disks are decreased, say \( D_1 \) and \( D_2 \). In other words, we consider region \( D_1 \cap D_2 \cap D_3 \). Let us introduce a point \( j_{21} \) as the symmetrical point for \( j_2 \) via the line \( j_1, j_2 \). Denote by \( T \) and \( D_j \) a triangle and a disk which are symmetrical to \( T \) and \( D_3 \) via the line \( j_1, j_2 \), respectively. In order to eliminate the triple intersection, we have to open new facility in \( T \). To save pairwise intersections and disk \( D_3 \), we have to exclude from \( T \) regions \( D_3, D'_1, \) and \( D'_2 \) (see Fig. 4). Note that symmetrical region in \( T \) has the same properties. Finally, we should remove interval \( j_1, j_2 \) from the resulting region.

**Case 3c:** New facility of the leader is opened in such a way that only one disk is decreased, say \( D_1 \). Denote by \( R \) the distance from \( j_1 \),
to region $D_2 \cap D_3$. Let us consider a disk $D''_1$ with radius $R$ and center in $j_1$. Each point from region $D''_1 \setminus (D_2 \cup D_3)$ eliminates the triple intersection but saves the disks $D_2$ and $D_3$ (see Fig. 5). In order to save pairwise intersections $D_1 \cap D_2$ and $D_1 \cap D_3$, we should exclude the disk $D''_1$ from the region (see case 2).

Let us consider the final structure of regions (see Fig. 6). They are described by the following points, intervals, and disks:

- three points $j_1, j_2, j_3$ (case 0);
- three intervals for pairs $(j_1, j_2), (j_2, j_3), (j_1, j_3)$ (case 1);
- six disks $D_j, D'_j, j = 1, 2, 3$ (case 2);
- six disks $D_1, D_2, D_3$ and their reflections $\overline{D_1}, \overline{D_2}, \overline{D_3}$ (cases 3a, 3b);
- three disks $D''_j, j = 1, 2, 3$ (case 3c).

We have at most 11 different instances of the follower problem from the triple of disks. The first one is the same as initial one, when the leader opens new facility outside of the grey area. The second one appears by deleting triple intersection and saving pairwise intersections. Three instances correspond to deleting only one pairwise intersection, other three instances correspond to deleting two pairwise intersections. The last three instances we have by opening new facility in one of the three points $j_1, j_2, j_3$.

Now let us consider all clients. The cases presented above give us all points, intervals, and disks which describe the boundaries of regions with constant goal function values. If we move new facility from one point to another one inside the same region, we have the same follower problem. The instance could change only if we change the region. As we remember, the leader facility is preferred in case of ties. Hence, each point on the boundary is not worse than every point inside of region. Thus, we can exploit points of intersections for the boundaries only and calculate the goal function value in each of them. We have at most $O(n^3)$ intervals and disks, and at most $O(n^3)$ of intersection points. Thus we get the desired. □

5. Local search

We use the obtained results for the local search under the Swap neighborhood. We apply the framework of the Variable Neighborhood Search (VNS, [16]), where $(k, l)$-Swap neighborhoods are used with different values $k$ and $l$. In these neighborhoods we move $k$ facilities of the leader to new positions but not far than the distance $l$ from the current positions. The values $l_i = 50, i = 2, \ldots, l_{\text{max}}$ and $k = 1, \ldots, k_{\text{max}}$ are used at the shaking step and $l_i = 50, k = 1$ at the local improvement step of the method. Below we present the pseudocode of the VNS algorithm for the $(r|p)$-centroid problem.

**VNS algorithm.**

*Initialization.* Find an initial solution $X$ of the leader and its market share $F(X)$; choose parameters $l_{\text{max}}, k_{\text{max}}$, and a stopping condition.

*Repeat* the following until the stopping condition is met:

1. $i \leftarrow 1; k \leftarrow 1$;
2. Repeat the following steps until $i \leq l_{\text{max}}$ and $k \leq k_{\text{max}}$:
   a. *Shaking:* Generate a solution $X'$ from the $(k, l_i)$-Swap neighborhood at random;
   b. *Local search:* Apply a local improvement method with $X'$ as initial solution; denote $X''$ the so obtained local optimum.
   c. *Move or not.* if $F(X) < F(X'')$, then $X \leftarrow X''$, $i \leftarrow 1$, $k \leftarrow 1$, else $i \leftarrow i + 1$; if $i > l_{\text{max}}$ then $i \leftarrow 1$, $k \leftarrow k + 1$.

As the stopping condition we use the running time of the method. The initial solution is generated by the alternating heuristic with the clustering procedure [10]. Step 2(b) of the method is the most time consuming. In order to reduce the running time of this Step, we apply two ideas. First of all, we divide the $(k,l)$-Swap
neighboring area into some subneighborhoods. Each subneighborhood contains the intersections of at most two types of lines:

- the intervals \( j_1, j_2 \) for \( j_1, j_2 \in J \);
- the disks \( D_j \) for \( j \in J \);
- the disks \( D_j \), \( D'_j \) for \( j \in J \);
- the disks \( D_j \) for \( j \in J \);
- the disks \( D'_j \) for \( j \in J \).

We investigate these subneighborhoods sequentially and apply the first improvement rule [17] for the local search at the Step 2(b) of the method. The second idea deals with the randomization of the neighborhood [1]. Instead of the search through all neighboring solutions we use a randomized neighborhood which contains each solution from the \( (k,l)\)-Swap neighborhood with a given probability. Moreover, we compute an upper bound for the leader market share instead of exact value for these neighboring solutions. These tricks allow us to reduce the computational efforts significantly without loss of quality for the final solution.

6. Computational experiments

We have coded the VNS algorithm in Delphi 7.0 environment and tested it on benchmark instances from the electronic library Discrete Location Problems. For all instances we have \( n=50 \), and demand points are randomly distributed among the square \( 7000 \times 7000 \) uniformly. Two types of weights are considered: \( w_j=1 \) and \( w_j \in [1,200] \). For all instances the behavior of the algorithm with \( p=r=10 \) is studied.

In the first computational experiment we try to see and understand the structure of the objective function of the \( (rXp, 1)\)-centroid problem. It is the market share of the leader. Hence, we will see a collection of plateaus. Moreover, in some plateaus we can discover some peaks. For example, in the case of four clients with the same weights \( w_j=1 \) and \( p=r=1 \), we have the following landscape (see Fig. 7).

The leader has no clients if his facility is outside of the parallelepiped. He has two clients if his facility is in the center of the parallelepiped and he has only one client if his facility is in another point between clients. In this illustrative example we observe two plateaus and one peak.

Fig. 8 shows the objective function for the randomly generated test instance, \( w_j \in [1,200] \). We observe a lot of peaks with different objective function values. Figs. 9 and 10 show the same landscape but from other points of view. Finally, Fig. 11 shows the landscape from the top point. We can see some disks, triangles, and their intersections. The central region is the most promising for the leader. But finding the optimal location for new facility is not trivial.

In the second experiment we check the size of \( (k,l)\)-Swap neighborhood for \( l=6 \) and \( k=1 \) and show the cardinalities of its subneighborhoods. We consider the same instance and open nine leader facilities according to the best known solution. Table 1 shows the number of elements in the subneighborhoods. For points \( j, j' \) we use the following notations:

- \( N_j \) is the number of mutual intersections for the intervals \( D_j, D'_j \);
- \( N_k \) is the number of mutual intersections for disks \( D_j, D'_j \);
- \( N_l \) is the number of intersections for disks \( D_j, D'_j \);
- \( N_m \) is the number of intersections for disks \( D_j, D'_j \) with disks \( D_j \);
- \( N_n \) is the number of intersections for disks \( D_j, D'_j \) with disks \( D'_j \);
- \( N_o \) is the number of intersections for disks \( D_j, D'_j \) with disks \( D'_j \).

Note that we ignore all disks \( D_j, j \in J \). Without loss of generality, we may drop intersections of the disks with other disks and
Table 1 indicates that the total number of neighboring solutions is $1\,363\,931 (i=500)$, but we have only 469 neighbors for $i=1$ and 1720 neighbors for $i=2$. Thus, we can use local improvement by the (1,1)-Swap neighborhood and have to apply randomization for $i, k > 1$. It is interesting to note that intervals generate a small part of neighboring solutions. In [10] the intervals are used for improvement only. Hence, the local search is a more powerful approach and can improve the solution considerably. We guess that some subneighborhoods can be reduced but this is a line for further research.

In the third computational experiment we test the VNS algorithm. We conducted the experiments in the PC Intel Xeon X5675, 3 GHz, RAM 96 GB, running under the Windows Server 2008 operating system. Table 2 shows the computational results for 10 instances and $w_j \in [1, 200]$. For each instance we run the algorithm with time limit 3 h. The second column of the Table 2 presents the market share of the leader according to the alternating heuristic from [7]. In brackets these values are shown as percentages. The third column shows computational results for alternating heuristic with clustering [10]. The last column presents the leader market share for the VNS algorithm. Table 3 presents computational results for identical client demands, $w_j = 1$ for all $j$. As we can see, the local search approach based on the discretization result for the $(r|x_p - 1)$-centroid problem is useful and can improve the leader market share.

In [11] we can find computational results for the discrete $(r|p)$-centroid problem with the same demands of the clients. As we can see, the leader market share exceeds the half of the market in discrete case. For the plane we have got 34–42% only. As it

<table>
<thead>
<tr>
<th>Instance code</th>
<th>Alternating heuristic</th>
<th>Procedure of clustering</th>
<th>VNS algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>111</td>
<td>1404 (31%)</td>
<td>1671 (37%)</td>
<td>1925 (42%)</td>
</tr>
<tr>
<td>211</td>
<td>1591 (28%)</td>
<td>1992 (35%)</td>
<td>2270 (40%)</td>
</tr>
<tr>
<td>311</td>
<td>1541 (29%)</td>
<td>1756 (37%)</td>
<td>1977 (41%)</td>
</tr>
<tr>
<td>411</td>
<td>1418 (31%)</td>
<td>1917 (36%)</td>
<td>2164 (41%)</td>
</tr>
<tr>
<td>511</td>
<td>1234 (23%)</td>
<td>1735 (38%)</td>
<td>1874 (42%)</td>
</tr>
<tr>
<td>611</td>
<td>1512 (27%)</td>
<td>1918 (34%)</td>
<td>2246 (41%)</td>
</tr>
<tr>
<td>711</td>
<td>1318 (26%)</td>
<td>1803 (36%)</td>
<td>1971 (40%)</td>
</tr>
<tr>
<td>811</td>
<td>1375 (27%)</td>
<td>1875 (37%)</td>
<td>2222 (42%)</td>
</tr>
<tr>
<td>911</td>
<td>1467 (29%)</td>
<td>1875 (37%)</td>
<td>2125 (42%)</td>
</tr>
</tbody>
</table>

Table 3

<table>
<thead>
<tr>
<th>Instance code</th>
<th>Alternating heuristic</th>
<th>Procedure of clustering</th>
<th>VNS algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>111</td>
<td>14 (28%)</td>
<td>16 (32%)</td>
<td>18 (36%)</td>
</tr>
<tr>
<td>211</td>
<td>13 (26%)</td>
<td>17 (34%)</td>
<td>18 (36%)</td>
</tr>
<tr>
<td>311</td>
<td>13 (26%)</td>
<td>16 (32%)</td>
<td>18 (36%)</td>
</tr>
<tr>
<td>411</td>
<td>12 (24%)</td>
<td>17 (34%)</td>
<td>19 (38%)</td>
</tr>
<tr>
<td>511</td>
<td>14 (28%)</td>
<td>17 (34%)</td>
<td>17 (34%)</td>
</tr>
<tr>
<td>611</td>
<td>15 (30%)</td>
<td>17 (34%)</td>
<td>18 (36%)</td>
</tr>
<tr>
<td>711</td>
<td>15 (30%)</td>
<td>16 (32%)</td>
<td>18 (36%)</td>
</tr>
<tr>
<td>811</td>
<td>15 (30%)</td>
<td>17 (34%)</td>
<td>19 (38%)</td>
</tr>
<tr>
<td>911</td>
<td>14 (28%)</td>
<td>16 (32%)</td>
<td>19 (38%)</td>
</tr>
</tbody>
</table>

Please cite this article as: Davydov I, et al. A local search heuristic for the $(r|p)$-centroid problem in the plane. Computers and Operations Research (2013), http://dx.doi.org/10.1016/j.cor.2013.05.003
is mentioned in [5], the continuous location problems as a rule are harder than discrete ones. Table 1 confirms this observation. Moreover, in the plane the follower has more opportunities to attack the leader facilities. As a result, the leader market share is small enough. Nevertheless, we guess that our computational results can be further improved and the leader can increase own market share.

7. Conclusions

We have considered the (rjp)-centroid problem on the Euclidean plane and developed the local search algorithm based on the VNS framework. It is known that the problem is $2^r$-hard and we have to solve the NP-hard follower problem in order to calculate the objective function value for a given solution of the leader. Our main theoretical result deals with the swap neighborhood for the leader solutions. We have shown that the best neighboring solution can be found in polynomial time for fixed $r$. Computational results for small test instances indicate that the problem is difficult indeed. The landscape of the $(rX_{p-1}+1)$-centroid subproblem is sophisticated. Finding the best neighboring solution is time consuming procedure. For future research it is interesting to find a way for accelerating the search process. Moreover, it is interesting to get an upper bound for the global maximum and design an exact method.

References