ISSN 0005-1179, Automation and Remote Control, 2014, Vol. 75, No. 4, pp. 677–687. © Pleiades Publishing, Ltd., 2014.
Original Russian Text © I.A. Davydov, Yu. A. Kochetov, N. Mladenovic, D. Urosevic, 2014, published in Avtomatika i Telemekhanika, 2014, No. 4, pp. 106–119.

Fast Metaheuristics for the Discrete $(r|p)$-Centroid Problem

I. A. Davydov*,, Yu. A. Kochetov*,**, N. Mladenovic***, and D. Urosevic*****

**Novosibirsk State University, Novosibirsk, Russia*
***Sobolev Institute of Mathematics, Siberian Branch, Russian Academy of Sciences, Novosibirsk, Russia*
****Mathematical Institute of the Serbian Academy of Sciences and Arts, Belgrade, Serbia*
e-mail: vann.davydov@gmail.com, jkochet@math.nsc.ru, Nenad.Mladenovic@brunel.ac.uk, draganu@mi.sanu.as.rs

Received November 3, 2013

Abstract—Two players, the leader and his competitor, open facilities, striving to capture the largest market share. The leader opens p facilities, then the follower opens r facilities. Each client chooses the nearest facility as his supplier. We need to choose p facilities of the leader in such a way as to maximize his market share. This problem can be represented as a bilevel programming problem. Based on this representation, in this work we propose two numerical approaches: local search with variable neighborhoods and stochastic tabu search. We pay the most attention to improving the methods' efficiency at no loss to the quality of the resulting solutions. Results of numerical experiments support the possibility to quickly find an exact solution for the problem and solutions with small error.

DOI: 10.1134/S0005117914040080

1. INTRODUCTION

Facility location problems comprise a wide array of mathematical models in operations research. This is a fruitful topic for both theoretical and experimental studies and for applications that arise in placing facilities and warehouses, police stations and fire stations, first aid facilities, constructing wireless communication networks and so on [1]. In most such models, one person makes a decision aiming to maximize his profit or minimize the total costs of servicing clients. In this work, we consider a more complex situation when two persons (players) sequentially make facility location decisions. This is the field of competitive location problems. It was initiated in a pioneering work of Hotelling [2]. This field studies self-interested strategies of behavior for two players under the assumption that clients are located along a line (a highway, coastline and so on). Later, richer model appeared, interesting from the points of view of both economics and game theory and for methods of optimization and operations research. A survey of such models and results of the corresponding studies is given, for instance, in [3, 4] (see also [5]).

One of such problems, namely the centroid problem, was first studied by Hakimi [3]. Consider a finite set I of possible locations for the facilities and a finite set J of the locations of the clients. The matrix (d_{ij}), $i \in I$, $j \in J$, specifies distances from clients to facilities. The value w_j determines a player's profit in servicing client j. Players sequentially make decisions on opening facilities, aiming to maximize their profits. First, the first player (leader) enters the market and opens p facilities. Knowing his decision, the second player (follower) opens his r facilities. Each client chooses the nearest facility out of the $p + r$ facilities both players have opened as his supplier. As a result, the set of clients is divided into two parts: clients of the leader clients of the follower. The problem is to find the p locations for the leader's facilities that will give him maximal profit (market share) under the strongest possible response by the follower. At present, fundamental studies of this problem have been divided into three directions:

—discrete problems, when sets of clients and facilities are finite [6–8];

—problems on a network, when clients are located at vertices of a graph, and facilities can be opened at arbitrary points on its edges [9, 10];

—problems in a Euclidean space, when clients are represented by a finite set of points, and facilities can be located at arbitrary points or in a given bounded region [11].

Yet another case appears when the set of clients is not finite but rather is given by a certain region on a plane with a given distribution density, but in this work we will not consider it further.

It is known [12] that the centroid problem in each of these three cases is Σ_2^P-hard, and the follower's problem for a given solution of the leader is NP-hard in the strong sense even if matrix (d_{ij}) specifies Euclidean distances on a plane. We will further restrict our attention to the problem when sets I and J are finite and coincide with each other.

Despite the problem's high complexity status, there are both exact and approximate methods for solving it that rely upon its combinatorial nature. Methods of implicit enumeration [13], branch-and-bound techniques [14], and an iteration method based on sequentially growing the set of follower's solutions [6] guarantee that the global optimum is found but are very computationally intensive already for $|I| = |J| = 100$, $p = r \geqslant 10$. Heuristics based on solutions of the p-median problem, genetic algorithms, particle swarm algorithms, and local search algorithms have shown high accuracy for a relatively small number of iterations [7], but the complexity of each iteration still remains rather high. This is due primarily to the fact that computing the leader's objective function requires an exact solution for the follower's problem. Since this problem is NP-hard in the strong sense, and we have to solve it multiple times, high computational costs seem to be inevitable. In [15], greedy algorithms are used for the follower's problem to reduce complexity. In [16], Lagrange relaxations have been used to the same purpose. The work [6] relaxed conditions on the fact that variables are integer, and the follower's problem was formulated as a linear programming problem. The resulting upper bound on the follower's profit was then used in local transformations of the leader's solutions. In [17], a particle swarm optimization algorithm was used instead of linear programming to solve the follower's problem.

In this work, we make another attempt to reduce the complexity of local search methods. We study two schemes: variable neighborhood search (VNS) and stochastic tabu search (STS). The main idea is to adapt these well known schemes to the considered bilevel programming problem. Both schemes use a neighborhood *Swap* for the leader's variables. The local search is made faster by dividing this neighborhood into three parts. Two of them are most promising and are searched through first of all. Since they are relatively small, we can quickly find the direction of ascent and reduce the complexity of one iteration. To solve the follower's problem we use exact methods or linear programming with metaheuristics, which also leads to reducing its complexity. Finally, in both methods the local search is randomized in order to diversify local search. As a result, we find efficient methods for solving the centroid problem that let us find exact solutions for $p = r \leqslant 15$ by an order of magnitude faster and for the first time find solutions for $p = r = 25$ and $p = r = 30$, a feat previously impossible due to high computational costs. Moreover, for $p = r = 30$ these methods work faster on average than for $p = r = 15$.

The paper is organized as follows. In Section 2, we give an exact mathematical setting of the problem. In Section 3, we present the variable neighborhood search. Section 4 is devoted to the stochastic method of tabu search. Section 5 gives results of numerical experiments with the examples from the *Discrete Location Problems* electronic library (http://math.nsc.ru/AP/benchmarks/). Results of these experiments indicate that both methods quickly find known optimal solutions, and on test examples where the optimum is not known their solutions differ little from each other.

2. PROBLEM SETTING

We denote by X the set of facilities opened by the leader; by Y, the set of facilities opened by the follower. We denote by $d(j, X)$ the distance between client j and the nearest leader's facility. Similarly, $d(j, Y)$ is the distance to the nearest facility of the follower. We will say that a client j prefers Y if $d(j, Y) < d(j, X)$ and prefers X otherwise. We denote by $J(Y \prec X)$ the set of clients who prefer Y, i.e.,

$$J(Y \prec X) = \{j \in J \,|\, d(j, Y) < d(j, X)\}.$$

Then the follower's profit $W(Y \prec X)$ is defined as the sum of profits received from all his clients:

$$W(Y \prec X) = \sum_{j \in J(Y \prec X)} w_j.$$

For a given solution X, the follower aims to maximize his profit. The value of his profit $W^*(X)$ is defined as the solution of the following problem:

$$W^*(X) = \max_{Y, |Y|=r} W(Y \prec X).$$

We call it the follower's problem, or the medianoid problem [3]. The leader aims to maximize his own profit or, which is the same, minimize the follower's profit. The minimal value $W^*(X^*)$ of the leader's losses is defined as the solution of the following problem:

$$W^*(X^*) = \min_{X, |X|=p} W^*(X).$$

The best solution of the leader X^* defines its profit $\sum_{j \in J} w_j - W^*(X^*)$. In the $(r|p)$-centroid problem we have to find X^* and $W^*(X^*)$.

We represent this minimax problem in terms of bilevel integer programming. We introduce the following variables:

$$x_i = \begin{cases} 1, & \text{if the leader opens facility } i \\ 0 & \text{otherwise,} \end{cases}$$

$$y_i = \begin{cases} 1, & \text{if the follower opens facility } i \\ 0 & \text{otherwise,} \end{cases}$$

$$z_j = \begin{cases} 1, & \text{if client } j \text{ is serviced from the leader's facilities} \\ 0, & \text{if client } j \text{ is serviced from the follower's facilities.} \end{cases}$$

Then $X = \{i \in I | x_i = 1\}$, $Y = \{i \in I | y_i = 1\}$. For each client j, we define the set of facilities $I_j(X)$ that lets the follower "capture" client j, i.e.,

$$I_j(X) = \left\{ i \in I \,\middle|\, d_{ij} < \min_{l \in I} \{d_{lj} | x_l = 1\} \right\}.$$

Note that if the distance to facilities of the leader and the follower is equal, the client will prefer the leader [3]. In the notation introduced above, the $(r|p)$-centroid problem can be represented as follows [6, 8]:

$$\max_x \sum_{j \in J} w_j z_j^*(X),$$

$$\sum_{i \in I} x_i = p,$$

$$x_i \in \{0, 1\}, \quad i \in I,$$

where $z_j^*(X), y_i^*(X)$ is the optimal solution for the follower's problem:

$$\max_{y,z} \sum_{j \in J} w_j(1 - z_j),$$

$$\sum_{i \in I} y_i = r,$$

$$1 - z_j \leqslant \sum_{i \in I_j(X)} y_i, \quad i \in I,$$

$$y_i, z_j \in \{0, 1\}, \quad i \in I, j \in J.$$

The objective function $W^*(X) = \sum_{j \in J} w_j z_j^*(X)$ defines the total profit of the leader given that he opens exactly p facilities. This value depends on the follower's optimal solution. The sets $I_j(X)$ for all $j \in J$ in this problem are assumed to be known. The objective function $W(Y \prec X) = \sum_{j \in J} w_j(1 - z_j)$ defines the follower's profit. The first constraint allows the follower to open exactly r facilities. The second constraint does not let him service client j if his facilities are not present in the set $I_j(X)$.

Note that the condition that variables z_j are integer can be replaced with condition $z_j \geqslant 0$ for all $j \in J$. The optimal value of the objective function in the follower's problem will remain unchanged. To solve it exactly one can use, for instance, the branch-and bound method.

A representation of the problem in terms of bilevel programming indicates a natural way to solve it: we restrict the search with the space of the leader's variables, computing his profit $W^*(X)$ with one method or another. Since this problem is close to the p-median problem in its structure, it is natural to use the method that has proven to be efficient in that problem [18].

3. VARIABLE NEIGHBORHOOD SEARCH

Metaheuristics are a powerful tool for solving complex combinatorial optimization problems. Their efficiency has been demonstrated with routing, location, and clustering problems, scheduling problems etc. [19]. Metaheuristics are general schemes for constructing heuristic algorithms that combine elements of randomization and learning, intensifying and diversifying the search, adequate control mechanisms, constructive heuristics, and local search techniques. Metaheuristics include simulated annealing approaches, tabu search, genetic algorithms, variable neighborhood search, ant colony optimization, and other techniques. In this section, we present one local search metaheuristic, namely variable neighborhood search. Its advantages include easy adaptation to complex mathematical models, e.g., load balancing models for web servers, where one has to predict user activity taking into account both the multicriterial nature of the load and a possibility to migrate web sites between servers.

Variable neighborhood search has been proposed as a general approach to combinatorial optimization problems by N. Mladenovic and P. Hansen [20]. Its main idea is to systematically vary the neighborhoods and the corresponding change in the landscape during the search for a global optimum. This method relies on the following three ideas:

—a local optimum with respect to one neighborhood is not necessarily also a local optimum with respect to another neighborhood;

—the global optimum is a local optimum with respect to any neighborhood;

—in many combinatorial optimization problems, local optima with respect to one or several neighborhoods are located rather close to each other.

The last fact is an empirical one; it is often observed in problems on the Euclidean plane [8]. This fact lets us narrow down the search region by using information about already found local optima. This idea underlies crossover operators for genetic algorithms, path relinking methods, ant colony optimization, and other approaches.

In application to the $(r|p)$-centroid problem, we introduce three neighborhoods to solve X that together will comprise a known neighborhood $Swap(X)$. By definition, it consists of all solutions of the leader obtained by closing one facility and opening another. Its size is $p(|I| - p)$. We divide it into three parts as follows.

Neighborhood $Fswap(X)$ consists of all leader's solutions that result from X by closing one facility and opening one facility from the solution $Y^*(X)$. The size of this neighborhood is pr.

Neighborhood $Nswap(X)$ consists of all leader's solutions that result from X by closing one facility and opening another, located at no more than the lth facility nearest to it. The size of this neighborhood is pl, where l is the neighborhood's parameter.

Neighborhood $Cswap(X)$ is the complement of these two neighborhoods,

$$Cswap(X) = Swap(X) \setminus \Big(Fswap(X) \cup Nswap(X)\Big).$$

The idea of this partition is to find, in a large neighborhood, small most promising parts that let us quickly find the best neighboring solution. Previous studies have shown that in the optimal solution, the follower's facilities are often in the immediate vicinity of the leader's facilities [7]. The follower attempts to "cut out" the leader from his clients. Due to this property, distinguishing neighborhoods $Fswap$ and $Nswap$ is in essence an attempt to predict the follower's behavior and find the strongest preventive move for the leader.

The method's general scheme looks as follows. On each iteration, we apply to the leader's current solution X a probabilistic procedure $Shake(X, k)$ that replaces k randomly chosen facilities of the leader by other randomly chosen facilities. To the resulting solution X', we apply a local improvement procedure first with respect to neighborhoods $Fswap$ and $Nswap$, and then, if necessary, with respect to the neighborhood $Cswap$. The resulting local optimum X'' is compared to the solution X. If the new local optimum is better than the previous one, we pass to the new local optimum. Otherwise, we change the parameter k and go to the next iteration. Here is the VNS method in pseudocode.

VNS method.

```
1. Find initial solution $X$, define the parameter $k_{\max}$.
2. Repeat until the stopping criterion is met:
    2.1. $k \leftarrow 1$,
    2.2. repeat until $k \leqslant k_{\max}$:
        (a) $X' \leftarrow Shake(X, k)$;
        (b) $X'' \leftarrow LocalSearch(X')$;
        (c) if $W^*(X'') > W^*(X)$ then $(X \leftarrow X'', k \leftarrow 1)$ else $k \leftarrow k + 1$.
3. Output $X$ as the result of the algorithm.
```

On step 1, as the initial solution we have selected an approximate solution of the p-median problem with a weighted distance matrix $(w_j d_{ij})$. In this case, the leader ignores the follower but tries to place his facilities as close to his product's customers as possible. This approach yields a good approximate solution [7]. As the stopping criterion, we use the number of transitions into an adjacent solution during local improvement or the computation time. Step 2.2(b) is the most computationally intensive. The efficiency of the entire approach mainly depends on its implementation. The *first improvement rule* let us stop investigating a neighborhood and pass to the next iteration as soon as we have found, for the current solution, an adjacent solution with a better value of the objective function. Since we go through the neighborhoods $Fswap$ and $Nswap$ first of all (first

$Fswap$, then $Nswap$), one often does not have to check the main part, $Cswap$. Besides, when we compute the objective function for adjacent solutions, we solve a linear programming problem instead of the original NP-hard follower's problem, which also reduces the computational cost of one iteration. Nevertheless, on step 2.2(c) we have to compute the value of $W^*(X'')$ exactly. To this purpose, we use the branch-and bound method (as implemented in CPLEX software). The best resulting solution (local maximum) is output as the result of the algorithm.

4. LOCAL TABU SEARCH

The tabu search method has been proposed by Fred Glover. It belongs to the class of trajectory metaheuristics and has been widely used to solve hard combinatorial optimization problems [21]. The method is based on an original local search scheme that lets one "travel" from one optimum to another in search for a global one rather than stop at a local optimum point. The main mechanism that lets us get out of local optima is a tabu list. This list contains solutions from previous iterations or fragments of such solutions (edges of the graph, colors of vertices, indices of replaced facilities and so on). With the tabu list, we remove a part of adjacent solutions from the neighborhood of the current solution and move to the best solution out of the remaining ones.

Together with the tabu list, randomizing the neighborhood also plays an important part. It lets us avoid looping, significantly reduces the time per iteration, and improves search efficiency. We denote by $Swap_q(X)$ the part of the neighborhood $Swap(X)$ chosen at random. Each element of the sets $Swap(X)$ is included in the set $Swap_q(X)$ independently of other elements with nonzero probability q. The set $Swap_q(X)$ may turn out to be empty or may coincide with the entire set $Swap(X)$. Under certain restrictions on the length of the tabu list, local search with this randomized neighborhood can asymptotically find the exact solution of the problem [16]. The previous method with varying neighborhoods also possesses similar properties [22]. Below we show a description of this method and adapt its main elements to solve the leader's problem. Below we show the pseudocode of the STS approach.

STS method.

```
1. Find initial solution $X$, define the randomization parameter $q$, construct an empty tabu list.
2. Repeat until the stopping criterion is met.
    2.1. Construct neighborhood $Swap_q(X)$ and remove forbidden elements from it.
    2.2. If the neighborhood is empty, return to step 2.1.
    2.3. Find an adjacent solution $X'$ with the largest estimate of the leader's profit.
    2.4. Let $X \leftarrow X'$ and compute the exact value of $W^*(X)$.
    2.5. Update the tabu list.
3. Show the best solution found.
```

The initial solution is chosen as before, by solving the p-median problem. The randomization parameter q for the neighborhood is set to be sufficiently small and does not change as the number of iterations grows. Neighborhood $Swap_q$ on step 2.3 is looked through in the same order as in the previous algorithm: first $Fswap$, then $Nswap$, and only then $Cswap$, if necessary. As soon as we have found an adjacent solution with a larger estimate of the objective function's value than the current solution, we stop going through the neighborhood. As the tabu list, we use an ordered list of pairs of the leader's facilities that have been closed and opened over the last few iterations. The length of the tabu list changes in a given interval during local search. If the best found solution begins to repeat itself, we increase the tabu list length by one; otherwise, we reduce it by one. On step 2.4 we use the CPLEX suite to compute the exact value of $W^*(X)$. The method stops after a given number of iterations are over or after a certain computation time has elapsed.

Step 2.3 is the most computationally intensive. For each adjacent solution, we have to estimate the value of the leader's objective function, i.e., solve the follower's problem. For this purpose, we again use stochastic local search with tabus. As the starting solution, we take the optimal solution of the follower's problem $Y^*(X)$ for the current leader's solution, which lets us restrict ourselves to a small number of iterations. We again use a randomized neighborhood $Swap_q$, but this time for the follower's solutions. Data structures proposed by Resende and Werneck [23] for the p-median problem can be adapted for the follower's problem, which significantly reduces the time needed to estimate the leader's profit.

5. EXPERIMENTAL STUDIES

Both methods have been implemented and have been tested on the examples from the *Discrete Location Problems* electronic library located at the server of the Institute of Mathematics of the Siberian Branch of the RAS. All samples have the same dimension $|J| = |I| = 100$. They are divided in two classes: *Euclidean* and *Uniform*. In the first class, the matrix (d_{ij}) specifies Euclidean distances between points on a plane, uniformly selecting 100 points in a square with size 7000. The values d_{ij} correspond to distances between points i and j, and the sets I and J coincide. In the second class, each element of the matrix (d_{ij}) is chosen independently of other elements uniformly at random in the interval from 0 to 10^4. In each class, we considered two types of profits. In the first case, $w_j = 1$ for all $j \in J$. In the second case, these values were chosen independently, uniformly at random in the interval from 1 to 200.

The first numerical experiment was designed to test the efficiency of the developed methods on test examples with known exact solutions. For the Euclidean class, such examples exist for $p = r \leqslant 15$. For $p = r = 20$, only record values are known together with estimates of how far they are from the optimum. For the second class of examples with uniform distribution of elements d_{ij}, optimal solutions are only known for $p = r = 7$. For large values of parameters, exact methods have too high computational costs.

Tables 1–4 show experimental results and the time needed to solve the problem in both methods. As the stopping criterion we used computational time. On each example, we used 5 minutes on a Pentium Intel Core Dual PC, 2.66 GHz, 2Gb RAM. The computation time in the tables corresponds to the moment when the best solution was obtained. This solution usually turned out to be optimal. If the optimal solution was not found, the resulting approximate solution had an error of less than one percent. Comparing these results with the best of already known results, the tabu search method with Lagrange relaxations (TSL) [16], we can note that the new approach indeed finds optimal solutions faster, especially for large values of the parameters p and r. In particular, for

Table 1. Results for the *Euclidean* class, $p = r = 10$

Inst	Optimum	W_{VNS}	t_{VNS}	W_{STS}	t_{STS}	W_{TSL}	t_{TSL}
111	4361	4361	20.7	4361	63.71	4361	342
211	5310	5310	24.47	5310	23.47	5310	405
311	4483	4483	20.41	4483	33.81	4483	819
411	4994	4994	20.72	4994	19.26	4994	270
511	4906	4906	200.38	4906	27.18	4906	396
611	4595	4595	107.44	4595	44.51	4595	504
711	5586	5586	56.56	5586	101.0	5586	207
811	4609	4609	207.56	4609	88.28	4609	603
911	5302	5302	24.9	5302	19.17	5302	297
1011	5005	5005	213.58	5005	103.54	5005	333
Avg	4915.1	4915.1	89.672	4915.1	52.39	4915.1	417

Table 2. Results for the *Euclidean* class, $p = r = 15$

Inst	Optimum	W_{VNS}	t_{VNS}	W_{STS}	t_{STS}	W_{TSL}	t_{TSL}
111	4596	4596	297.55	4596	173.28	4596	545
211	5373	5373	200.47	5373	88.92	5373	12021
311	4800	4800	22.55	4800	91.1	4800	3126
411	5064	5058	110.35	5064	121.3	5064	1442
511	5131	5123	193.88	5131	216.23	5123	1928
611	4881	4881	83.18	4881	114.77	4881	781
711	5827	5827	280.44	5827	210.91	5827	2043
811	4675	4620	301.67	4675	123.41	4675	1088
911	5158	5157	253.84	5158	157.8	5158	1322
1011	5195	5195	22.35	5195	48.19	5195	2403
Avg	5070	5063	176.628	5070	134.59	5069.2	2669.9

Table 3. Results for the *Euclidean* class, $p = r = 20$

Inst	BestKnown	W_{VNS}	t_{VNS}	W_{STS}	t_{STS}	W_{TSL}	t_{TSL}
111	4512	4481	274.55	4484	118.14	4512	6008
211	5432	5432	245.89	5432	289.15	5432	4922
311	4893	4893	115.61	4893	211.32	4893	9365
411	5209	5209	234.56	5209	288.86	5209	5165
511	5334	5334	120.38	5334	133.18	5334	7922
611	4952	4944	234.46	4944	198.07	4952	13 081
711	5893	5893	126.68	5893	254.25	5893	6000
811	4858	4858	285.74	4858	118.8	4858	8526
911	5459	5455	273.42	5455	202.24	5459	1023
1011	5399	5399	166.26	5399	184.43	5399	2347
Avg	519.1	5189.8	207.755	5190.1	199.84	5194.1	6435.9

Table 4. Results for the *Uniform* class, $p = r = 7$

Inst	Optimum	W_{VNS}	t_{VNS}	W_{STS}	t_{STS}
123	5009	5009	304.17	5009	65.09
223	5459	5459	182.91	5459	63.18
323	5019	5009	145.01	5019	54.69
423	4908	4908	296.63	4908	145.22
523	5208	5198	292.05	5208	22.63
623	5032	5032	296.52	5032	197.08
723	5055	5055	286.04	5055	62.23
823	4951	4860	295.77	4951	74.49
923	5127	5060	217.7	5127	111.27
1023	5084	5067	322.48	5084	278.18
Avg.	5085.2	5065.7	263.928	5085.2	107.406

$p = r = 15$ average time of computation reduces by a factor of up to 15 even when the difference in computer performance is taken into account. Thus, the proposed methods are able to find the global optimum rather quickly.

The second computational experiment was designed to study the algorithms' behavior on the same test examples but with the hardest values of parameters p and r. As noted in [6, 7], these values correspond to a third of the size of the set I. Tables 5–7 show computation results for $p = r = 25$ and $p = r = 30$. Previously, computational experiments have not been conducted in such dimension. As the stopping criterion we used the number of iterations, i.e., the number of

Table 5. Results for the *Euclidean* class

	$p = r = 25$				$p = r = 30$			
Inst	W_{VNS}	t_{VNS}	W_{STS}	t_{STS}	W_{VNS}	t_{VNS}	W_{STS}	t_{STS}
111	4653	19.85	4653	27.46	4945	19.99	4945	29.41
211	5302	252.82	5301	64.12	5581	20.09	5581	34.18
311	5101	53.01	5101	113.82	5237	68.32	5273	61.15
411	5375	157.5	5375	104.18	5509	34.58	5509	54.28
511	5425	43.98	5425	48.67	5602	27.16	5541	46.81
611	5051	89.48	5051	22.61	5352	23.69	5352	44.52
711	5978	30.00	5978	64.18	6187	102.91	6195	108.61
811	5104	143.62	5104	54.6	5336	107.29	5336	18.45
911	5801	22.74	5801	212.81	5922	20.22	5922	27.46
1011	5465	62.22	5465	72.13	5646	34.06	5646	34.15
Avg.	5325.5	87.52	5325.4	78.45	5531.7	45.831	5530.0	45.9

Table 6. Results for the *Euclidean* class, $w_j = 1$

	$p = r = 25$				$p = r = 30$			
Inst	W_{VNS}	t_{VNS}	W_{STS}	t_{STS}	W_{VNS}	t_{VNS}	W_{STS}	t_{STS}
111	53	33.14	53	56.88	58	155.74	58	84.16
211	52	65.78	52	21.17	56	58.35	56	119.57
311	57	80.48	57	12.16	58	34.3	58	56.61
411	53	274.94	53	106.18	55	64.56	55	33.98
511	53	18.84	53	37.14	58	48.02	58	45.18
611	57	15.69	57	148.12	59	25.67	59	27.11
711	54	18.71	54	8.14	58	31.92	58	18.67
811	55	42.25	55	86.11	57	40.52	57	25.94
911	58	46.18	58	89.17	59	15.71	59	35.85
1011	55	39.645	55	64.81	59	18.24	59	67.94
Avg.	54.7	63.57	54.7	62.98	57.7	49.3	57.7	51.5

Table 7. Results for the *Uniform* class, $w_j = 1$

	$p = r = 25$				$p = r = 30$			
Inst	W_{VNS}	t_{VNS}	W_{STS}	t_{STS}	W_{VNS}	t_{VNS}	W_{STS}	t_{STS}
123	62	16.121	62	30.27	70	15.732	69	50.97
223	62	157.402	62	38.84	65	15.811	65	67.87
323	61	34.595	61	47.48	65	15.823	65	86.92
423	59	25.587	59	17.26	65	16.086	65	91.13
523	63	70.906	63	41.00	69	15.583	69	76.31
623	62	16.385	61	35.61	69	15.943	69	78.44
723	66	15.685	66	19.42	72	15.641	72	74.2
823	60	32.976	60	19.38	62	15.981	62	80.56
923	63	18.964	63	17.43	68	15.644	68	59.36
1023	65	15.49	65	21.58	73	15.421	71	126.19
Avg.	62.3	40.411	62.2	28.82	67.8	15.76	67.5	79.19

transitions from the current value to an adjacent one; we set this number to 1000. Since the global optimum is not known, the methods were compared with each other by comparing the best found solutions. The tables indicate that both methods demonstrate approximately the same efficiency, and computation results often coincide or differ insignificantly. Increasing the number of iterations does not cause the record values to grow. Since both methods are asymptotically exact, it appears that the resulting solutions are optimal, but it appears very hard to prove this claim.

Comparing Tables 2 and 5, one can note that the computation time, for instance, in tests 111, 511, 611, 711 drops as p and r grow. This may look odd since for $p = r = 25$ and $p = r = 30$ exact methods are already unable to find a solution for the problem. Nevertheless, local search methods find record values even faster than for $p = r = 15$. This can be explained by the fact that the size of set I remains unchanged. As the parameter p grows, the follower's problem reduces in dimension. The leader captures more and more better facilities, and the follower's problem simplifies. As a result, the total time required to find the record solution may decrease, but also may increase since the leader has more possibilities now.

6. CONCLUSION

We have considered a known Σ_2^P-hard $(r|p)$–centroid problem in the discrete setting. To solve the problem, we have developed two local search methods based on representing the problem as a bilevel discrete optimization problem. Since for a fixed upper level solution the problem remains NP-hard in the strong sense, we pay the most attention to reducing complexity of one local search iteration. We show that new methods are indeed much more efficient than their predecessors, and they let one quickly find optimal solutions or solutions with small relative error. In further studies, it will be interesting to find similar methods for problems on a network, when clients are located at the vertices of a graph, and facilities can be opened in arbitrary points on its edges. This version of the problem has only been studied from the point of view of its computational complexity. Numerical methods are virtually nonexistent, although for the follower's problem on the lower level there already exist efficient metaheuristics [24]. Nontrivial upper bounds for the optimum and exact methods for this problem are also unknown at present, and they are undeniably very interesting for further studies.

REFERENCES

1. *Facility Location. A Survey of Applications and Methods*, Drezner, Z., Ed., Berlin: Springer, 1995.
2. Hotelling, H., Stability in Competition, *Econ. J.*, 1929, vol. 39, pp. 41–57.
3. Hakimi, S.L., Locations with Spatial Interactions: Competitive Locations and Games, in *Discrete Location Theory*, Mirchandani, P.B. and Francis, R.L., Eds., London: Wiley, 1990, pp. 439–478.
4. Kress, D. and Pesch, E., Sequential Competitive Location on Networks, *Eur. J. Oper. Res.*, 2012, vol. 217, pp. 483–499.
5. Beresnev, V. and Mel'nikov, A., Approximate Algorithms for the Competitive Facility Location Problem, *J. Appl. Ind. Math.*, 2011, vol. 5, no. 2, pp. 180–190.
6. Alekseeva, E., Kochetova, N., Kochetov, Y., et al., A Heuristic and Exact Methods for the Discrete $(r|p)$-Centroid Problem, *LNCS*, 2010, vol. 6022, pp. 11–22.
7. Alekseeva, E. and Kochetov, Y., Metaheuristics and Exact Methods for the Discrete $(r|p)$-Centroid Problem, in *Metaheuristics for Bi-Level Optimization*, Talbi, E.-G. and Brotcorne, L., Eds., Berlin: Springer, 2013.
8. Kochetov, Y.A., Facility Location: Discrete Models and Local Search Methods, in *Combinatorial Optimization. Methods and Applications*, Chvatal, V., Ed., Amsterdam: IOS Press, 2011, pp. 97–134.
9. Noltemeier, H., Spoerhase, J., and Wirth, H., Multiple Voting Location and Single Voting Location on Trees, *Eur. J. Oper. Res.*, 2007, vol. 181, pp. 654–667.
10. Spoerhase, J. and Wirth, H., (r, p)-Centroid Problems on Paths and Trees, *J. Theor. Comput. Sci. Archive*, 2009, vol. 410, pp. 5128–5137.
11. Davydov, I., Kochetov, Y., and Carrizosa, E., A Local Search Heuristic for the $(r|p)$-Centroid Problem in the Plane, *Comput. Oper. Res.*, 2013, DOI: 10.1016/j.cor.2013.05.003.

12. Davydov, I., Kochetov, Y., and Plyasunov, A., On the Complexity of the $(r|p)$-Centroid Problem on the Plane, *TOP*, 2013, DOI: 10.1007/s11750-013-0275-y.

13. Campos Rodríguez, C.M., Santos Peñate, D.R., and Moreno Perez, J.A., An Exact Procedure and LP Formulations for the Leader–Follower Location Problem, *TOP*, 2010, vol. 18, no. 1, pp. 97–121.

14. Roboredo, M.C. and Pessoa, A.A., A Branch-and-Cut Algorithm for the Discrete $(r|p)$-Centroid Problem, *Eur. J. Oper. Res.*, 2013, vol. 224, no. 1, pp. 101–109.

15. Benati, S. and Laporte, G., Tabu Search Algorithms for the $(r|X_p)$-Medianoid and $(r|p)$-Centroid Problems, *Locat. Sci.*, 1994, vol. 2, no. 2, pp. 193–204.

16. Davydov, I.A., Local Tabu Search for the Discrete $(r|p)$-Centroid Problem, *Diskret. Anal. Issled. Oper.*, 2012, vol. 19, no. 2, pp. 19–40.

17. Campos Rodríguez, C.M., Moreno Pérez, J.A., and Santos Peñate, D.R., Particle Swarm Optimization with Two Swarms for the Discrete $(r|p)$-Centroid problem, *LNCS*, 2012, vol. 6927, pp. 432–439.

18. Mladenović, N., Brimberg, J., Hansen, P., and Moreno-Pérez, J.A., The p-Median Problem: A Survey of Metaheuristic Approaches, *Eur. J. Oper. Res.*, 2007, vol. 179, pp. 927–939.

19. Talbi, E-G., *Metaheuristics: From Design to Implementation*, Berlin: Wiley, 2009.

20. Mladenović, N. and Hansen, P., Variable Neighbourhood Search, *Comput. Oper. Res.*, 1997, vol. 24, pp. 1097–1100.

21. Glover, F. and Laguna, M., *Tabu Search*, Dordrecht: Kluwer, 1997.

22. Brimberg, J., Hansen, P., and Mladenović, N., Attraction Probabilities in Variable Neighborhood Search, *4OR*, 2010, vol. 8, no. 2, pp. 181–194.

23. Resende, M.G.C. and Werneck, P.F., A Hybrid Heuristic for the p-Median Problem, *J. Heurist.*, 2004, vol. 10, pp. 59–88.

24. Roksandic, S., Carrizosa, E., Urosevic, D., and Mladenović, N., Solving Multifacility Huff Location Models on Networks Using Variable Neighborhood Search and Multi-Start Local Search Metaheuristics, *Electron. Notes Discret. Math.*, 2012, vol. 39, no. 1, pp. 121–128.

This paper was recommended for publication by A.I. Kibzun, a member of the Editorial Board