Facility Location: Discrete Models and Local Search Methods

Yury KOCHETOV
Sobolev Institute of Mathematics, Novosibirsk, Russia

Abstract. Discrete location theory is one of the most dynamic areas of operations research. We present the basic mathematical models used in this field, as well as their properties and relationship with pseudo-Boolean functions. We also investigate the theory of PLS-complete problems, average and worst case computational complexity of the local search algorithms, and approximate local search. Finally, we discuss computationally difficult test instances and promising directions for further research.

Keywords. local search, PLS–complete problems, pseudo–Boolean function, Karush–Kuhn–Tucker conditions, metaheuristics

Introduction

Facility location constitutes a broad spectrum of mathematical models, methods, and applications in operations research. It is an interesting topic for theoretical studies, experimental research, and real-world applications. Examples include storage facilities, warehouses, police and fire stations, base stations for wireless services, and others [1]. Who actually proposed the first mathematical model in this field will probably never be known. It is most common to credit Pierre de Fermat (1601–1665) and Evangelista Torricelli (1608–1647), who studied a basic form of the spacial median problem (see [2] for a historical review of the literature).

Surely no paper can cover all aspects of facility location. In these lecture notes, we discuss only the basic discrete models and present theoretical results for local search methods. In Section 1 we consider the well-known uncapacitated facility location problem and its generalizations. The main idea of this section is to show the useful relationship between facility location models and pseudo–Boolean functions. In Section 2 we introduce some simple neighborhoods and discuss the relations between local optima and classical Karush–Kuhn–Tucker conditions. In Section 3 we define the class PLS (polynomial time local search problems) and show that some local search problems in facility location are the most difficult in this class. In Section 4 we investigate the quality of local optima. More precisely, we introduce the class GLO (Guaranteed Local Optima) and show that the closure of GLO under PTAS reductions coincides with the class APX, which is the set of optimization problems with underlying decision problem in NP that allow polynomial-time constant-factor approximation algorithms. Finally, in Section 5 we discuss difficult test instances for the local search methods.

1. Discrete facility location models

1.1. The uncapacitated facility location problem

This problem (UFLP) has been also called *the simple plant location problem* in the early literature [3,4]. Its input consists of a finite set I of *sites* with nonnegative *fixed costs* f_i of opening a facility in site i and a finite set J of *users* with a nonnegative *production–transportation costs* c_{ij} of servicing user j from a facility opened in site i. The goal is to find a set of sites such that opening facilities there minimizes the total cost of servicing all users. This means finding a nonempty subset S of I, which minimizes the objective function F defined by

$$F(S) = \sum_{i \in S} f_i + \sum_{j \in J} \min_{i \in S} c_{ij} :$$

the first term is the fixed cost for opening facilities in all sites in S and the second term is the production-transportation cost for servicing all users from these sites.

This problem is NP-hard in the strong sense and it is difficult to approximate: unless P = NP, it admits no constant-factor polynomial-time approximation algorithm, and so it does not belong to the class APX. Polynomially solvable cases and approximation algorithms are described in [5,6].

For the metric case, when the matrix (c_{ij}) satisfies the triangle inequality, the problem is strongly NP-hard again and Max SNP-hard. The best approximation algorithm has a guaranteed performance ratio of 1.52 and is suggested in [7]. A 1.463 factor approximation algorithm would imply P = NP [8]. For the special case of the metric UFLP when facilities and users are points in d–dimensional Euclidean space and the production-transportation costs are geometrical distances between the points, an approximation scheme is suggested meaning an $(1+\varepsilon)$-factor approximation algorithm for each positive ε with running time polynomial in $|I|$ and $|J|$ and exponential in d and $1/\varepsilon$. An excellent review of different techniques for approximations in the metric case can be found in [9].

Exact branch and bound methods with lower bounds based on linear programming relaxations have been developed by research teams in several countries. In describing these bounds, we shall find it convenient to assume that $I = \{1, \ldots, m\}$ and $J = \{1, \ldots, n\}$. The first approaches used a *weak* linear 0–1 programming formulation [10]. Let us introduce the decision variables:

$$x_i = \begin{cases} 1 & \text{if facility } i \text{ is opened,} \\ 0 & \text{otherwise,} \end{cases}$$

$$x_{ij} = \begin{cases} 1 & \text{if user } j \text{ is serviced from facility } i, \\ 0 & \text{otherwise.} \end{cases}$$

Now the UFLP can be written as a 0–1 program:

$$\min \Big\{ \sum_{i \in I} f_i x_i + \sum_{j \in J} \sum_{i \in I} c_{ij} x_{ij} \Big\}$$

$$\text{subject to} \quad \sum_{i \in I} x_{ij} = 1, \quad j \in J,$$

$$n x_i \geq \sum_{j \in J} x_{ij}, \quad i \in I,$$

$$x_i, x_{ij} \in \{0, 1\}, \quad i \in I, j \in J.$$

If we replace the last restriction by $x_i, x_{ij} \in [0, 1]$ for all $i \in I, j \in J$ we see that every optimal solution satisfies

$$x_i = \frac{1}{n} \sum_{j \in J} x_{ij}, \quad i \in I$$

and a lower bound can be computed as the optimal value of the following trivial linear programming problem:

$$\min \sum_{j \in J} \sum_{i \in I} (f_i / n + c_{ij}) x_{ij}$$

$$\text{subject to} \quad \sum_{i \in I} x_{ij} = 1, \quad j \in J,$$

$$0 \leq x_{ij} \leq 1, \quad i \in I, j \in J.$$

We may compute the optimal solution for the problem easily, but this lower bound is not sharp. If we replace m restrictions $n x_i \geq \sum_{j \in J} x_{ij}$ by $n \times m$ restrictions

$$x_i \geq x_{ij}, \quad i \in I, j \in J,$$

we will get an equivalent reformulation of the UFLP with a better lower bound. Note that we do not have an analytical solution yet. This *strong* reformulation has been used in the exact methods by researchers of Russia [11,12], Ukraine [13], Scandinavia [14], and the USA [15] independently. Recently, the strong formulation was used for solving large scale instances as well [16,17]. Therefore, for combinatorial optimization problems we can find different equivalent reformulations and the choice of formulation is important.

There is a useful relationship between the UFLP and minimization problem for pseudo–Boolean functions. It was first noted by P. Hammer and S. Rudeanu [18,19]. Later, V. Beresnev [20] suggested another reduction of the UFLP to a minimization problem for pseudo–Boolean polynomial with positive coefficients for nonlinear terms. Moreover, it has been shown that these combinatorial optimization problems are equivalent. Below we discuss the reduction in details.

For a vector $g = (g_1, \ldots, g_m)$ with ranking

$$g_{i_1} \leq g_{i_2} \leq \ldots \leq g_{i_m},$$

we introduce a vector $\Delta g = (\Delta g_0, \ldots, \Delta g_m)$ in the following way:

$$\Delta g_0 = g_{i_1};$$

$$\Delta g_l = g_{i_{l+1}} - g_{i_l},\ 1 \leq l < m;$$

$$\Delta g_m = g_{i_m}.$$

Lemma 1 [11,20]. *For each 0-1 vector $z = (z_1, \ldots, z_m)$ distinct from $\{1, \ldots, 1\}$ the following equations hold:*

$$\min_{i|z_i=0} g_i = \Delta g_0 + \sum_{l=1}^{m-1} \Delta g_l z_{i_1} \ldots z_{i_l};$$

$$\max_{i|z_i=0} g_i = \Delta g_m - \sum_{l=1}^{m-1} \Delta g_{m-l} z_{i_{m-l+1}} \ldots z_{i_m}.$$

Let the ranking for column j of the matrix (c_{ij}) be

$$c_{i_1^j j} \leq c_{i_2^j j} \leq \ldots \leq c_{i_m^j j}$$

Using Lemma 1, we can get a pseudo–Boolean function for the UFLP:

$$b(z) = \sum_{i \in I} f_i(1 - z_i) + \sum_{j \in J} \sum_{l=0}^{m-1} \Delta c_{lj} z_{i_1^j} \ldots z_{i_l^j}.$$

Below we will see the relationship between the UFLP and the minimization problem for this real-valued function defined on the $2^n - 1$ points of the hypercube distinct from $(1, \ldots, 1)$.

Theorem 1 [11,20]. *The minimization problem for the pseudo–Boolean function $b(z)$ for $z \neq (1, \ldots, 1)$ and the UFLP are equivalent. For optimal solutions z^*, S^* of these problems we have $F(S^*) = b(z^*)$ and $z_i^* = 0 \Leftrightarrow i \in S^*$ for all $i \in I$.*

Let us consider an illustrative example. Put $I = J = \{1, 2, 3\}$,

$$f_i = \begin{pmatrix} 10 \\ 10 \\ 10 \end{pmatrix} \qquad c_{ij} = \begin{pmatrix} 0 & 3 & 10 \\ 5 & 0 & 0 \\ 10 & 20 & 7 \end{pmatrix}.$$

According to Lemma 1, the corresponding function $b(z)$ is the following:

$$b(z) = 10(1-z_1) + 10(1-z_2) + 10(1-z_3) + (5z_1 + 5z_1z_2) + (3z_2 + 17z_1z_2) +$$

$$(7z_2 + 3z_2z_3) = 15 + 5(1-z_1) + 0(1-z_2) + 10(1-z_3) + 22z_1z_2 + 3z_2z_3.$$

Let us try to reconstruct an instance of the UFLP: we obtain $I' = I, J' = \{1, 2\}$,

$$f'_i = \begin{pmatrix} 5 \\ 0 \\ 10 \end{pmatrix} \qquad c'_{ij} = \begin{pmatrix} 0 & 3 \\ 0 & 0 \\ 22 & 0 \end{pmatrix}.$$

The dimension of the new instance is less than the dimension of the original one, $|J'| < |J|$. Moreover, $f'_2 = 0$, hence, we may open the second facility without loss of optimality. In other words, we get a new equivalent instance of the UFLP with smaller dimension. Different instances of the UFLP can lead to the same function $b(z)$. Thus, we may try to reduce the dimension before solving the problem.

Theorem 2. [21] *For the minimization problem of the pseudo–Boolean function $b(z)$ with positive coefficients in the nonlinear terms, the equivalent instance of the UFLP with a minimal number of users can be found in polynomial time from n and m.*

Idea of proof. Consider an arbitrary pseudo–Boolean function $b(z)$ with positive coefficients in the nonlinear terms. Let L be the set of nonlinear terms and the function $b(z)$ defined by

$$b(z) = \sum_{i \in I} \alpha_i (1 - z_i) + \sum_{l \in L} \beta_l \prod_{i \in I_l} z_i, \quad \text{where} \quad \beta_l > 0 \text{ and } I_l \subset I \text{ for all } l \in L.$$

The family of subsets $\{I_l\}_{l \in L}$ of the set I with order relation $I_{l'} < I_{l''} \Leftrightarrow I_{l'} \subset I_{l''}$ forms a partially ordered set (poset). An arbitrary sequence of subsets $I_{l_1} < \ldots < I_{l_k}$ is called a *chain*. An arbitrary partition of the family $\{I_l\}_{l \in L}$ into nonoverlapping chains induces a matrix (c_{ij}) for the UFLP. Each element of the partition corresponds to a user. The requirement to find an instance of the UFLP with a minimal number of users is equivalent to finding a partition of the poset into the minimal number of nonoverlapping chains. This is a well-known problem, which can be solved in polynomial time (see Dilworth's Theorem [22]).

The minimization problem for $b(z)$ is equivalent to the UFLP but it has some new properties. Let us consider this problem for continuous variables z_i from the interval $[0, 1]$. In the UFLP this replacement leads to an integrality gap:

$$gap = (F(S^*) - F_{LP})/F(S^*),$$

where F_{LP} is the optimal value for the linear programming relaxation. It can be arbitrary close to 1 [23]. For the minimization problem of $b(z)$ the gap equals 0.

Theorem 3 [24]. *The set of optimal solutions of the minimization problem for arbitrary pseudo–Boolean function with continuous variables contains a pure integer solution.*

Suppose now that the fixed costs are the same for all facilities and that we open exactly p facilities. The first item in the objective function of the UFLP is a constant and we wish to minimize the objective function F defined by

$$F(S) = \sum_{j \in J} \min_{i \in S} c_{ij}$$

for every subset S from I with cardinality p. This problem is known as the discrete p-median problem. It is NP–hard in the strong sense and existence of a $2^{q(n,m)}$-factor approximation algorithm for a polynomial q would imply P=NP [25]. In other words, this problem does not belong to the class APX and a good approximate solution is hard to find, as is the optimal one. The p-median problem can be reduced to the minimization problem for a pseudo–Boolean function as well. However, now one additional restriction $\sum_{i \in I} z_i = m - p$ is added. Reformulations of Theorems 2 and 3 for the problem are valid as well.

1.2. The Multi–Stage Facility Location Problem

Let us consider a more complicated situation, where we need several facilities to produce the goods for users. We assume that there are k types of facilities (plants, warehouses, distribution centers and others) and we need facilities of all types to produce the goods. As in the previous model, the set of facilities I and the set of users J are finite. We assume that $I = I_1 \cup \ldots \cup I_k$, where I_l is the set of facilities of type l and $I_{l_1} \cap I_{l_2} = \emptyset$ whenever $l_1 \neq l_2$. Let P denote the set of all admissible facility paths. For each path p from P we know a sequence of facilities $p = \{i_1, \ldots, i_k\}$, where $i_l \in I_l$ for each l from 1 to k. For each facility i we have the following nonnegative parameters: c_i is the production cost, $d_{ii'}$ is the transportation cost between facilities i and i', and the set P_i of facility paths, which contain this facility. Denote by D_{pj} the total transportation cost for the facility path p and user j:

$$D_{pj} = d_{i_1 i_2} + \ldots + d_{i_{k-1} i_k} + d_{i_k j}.$$

Similarly, the total production cost for the facility path p is the following:

$$C_p = c_{i_1} + \ldots + c_{i_k}.$$

The amount of goods for user j we denote by φ_j. In the Multi–Stage Uncapacitated Facility Location Problem (MSUFLP) we need to find a subset of facilities and a subset of facility paths in such a way to service all users with minimal total cost for opening facilities, producing the goods and transporting them to users. Using similar variables as for the UFLP, we can write the problem in the following way:

$$\min \left\{ \sum_{i \in I} f_i x_i + \sum_{j \in J} \varphi_j \sum_{p \in P} (C_p + D_{pj}) x_{pj} \right\}$$

$$\text{subject to} \quad \sum_{p \in P} x_{pj} = 1, \quad j \in J,$$

$$\sum_{p \in P_i} x_{pj} \ \leq x_i, \quad i \in I, \ j \in J,$$

$$x_{pj}, \ x_i \in \{0, 1\}, \quad i \in I, \ p \in P, \ j \in J.$$

The objective function is the total cost. The first restriction ensures that all users are satisfied. The second restriction allows us to use opening facilities only for servicing the users.

The problem is strongly NP–hard and closely related to standardization and unification problems [11,14], pseudo–Boolean functions, and as we will see below, the bilevel facility location problems. If each facility path contains exactly one facility then we get the UFLP. Therefore, it is a difficult problem for approximation. For the metric case some approximation algorithms are developed in [26,27]. The branch and bound methods based on heuristics for the dual linear programming problems are studied in [28,29]. Polynomially solvable cases are described in [30,31].

The restriction $x_{pj} \ \leq x_i$ for all $p \in P_i, i \in I, \ j \in J$, can be used instead of

$$\sum_{p \in P_i} x_{pj} \ \leq x_i, \quad i \in I, \ j \in J.$$

It is easy to check that this replacement gives the same integer optimal solution but a weak linear programming relaxation. The same effect will occur (see [33]) if we introduce new variables $y_p = 1$ if the facility path p is used and $y_p = 0$ otherwise, and replace the restriction $\sum_{p \in P_i} x_{pj} \ \leq x_i$ for all $i \in I, \ j \in J$, by the following:

$$y_p \geq x_{pj}, \quad j \in J, \ p \in P,$$

$$x_i \geq y_p, \quad p \in P_i, \ i \in I.$$

Again, we have the same 0–1 optimal solution and a weak linear programming relaxation. Moreover, there is a family of instances for the MSUFLP where the ratio of optimal value of the original and new linear programming relaxations may be arbitrarily large. Let us rewrite the MSUFLP as an unconstrained problem. Define $C_{pj} = \varphi_j(C_p + D_{pj})$ for all $p \in P, \ j \in J$. Now the MSUFLP can be written as the minimization problem for the function F defined by

$$F(y) = \sum_{i \in I} f_i \max_{p \in P_i} \{y_p\} + \sum_{j \in J} \min_{p \in P} \{C_{pj} \mid y_p = 1\}.$$

Using Lemma 1 we can obtain the following pseudo–Boolean function:

$$B(z) = \sum_{i \in I} f_i (1 - \prod_{p \in P_i} z_p) + \sum_{j \in J} \sum_{l=0}^{|P|-1} \Delta C_{lj} z_{i_1^j} \dots z_{i_l^j}.$$

Note that this function has positive and negative nonlinear terms.

Theorem 4. [11,20] *The MSUFLP is equivalent to the minimization problem for pseudo–Boolean function $B(z)$ for $z \neq (1, \dots, 1)$. For optimal solutions z^*, y^* of these problems we have $B(z^*) = F(y^*)$ and $z_p^* = 1 - y_p^*$ for all $p \in P$.*

Therefore, the MSUFLP can be reduced to a minimization problem for a pseudo–Boolean function. For an arbitrary pseudo–Boolean function we can reconstruct an equivalent instance of the MSUFLP with the minimal number of users.

1.3. Facility location with user preferences

Thus far we have assumed that there was only one decision maker who tried to minimize the total cost of opening facilities and servicing users. However, users may be free to choose the facility. They may have their own preferences, for example, the travel time to a facility. They do not have to minimize the production and transportation costs of the firm. Hence, we should include user preferences in the mathematical model [32].

Let the matrix (g_{ij}) define the user preferences on the set I. If $g_{i_1 j} < g_{i_2 j}$, then user j prefers facility i_1. We assume for simplicity that all elements are different in each column of the matrix. Otherwise, we must consider cooperative and noncooperative strategies for the decision maker and users [34]. Therefore, the decision maker wishes to find a subset S of opening facilities in such a way that all users will be serviced with minimal total cost, taking into account user preferences. For this case, the mathematical model can be presented as the 0–1 bilevel linear programming problem [35,36]: minimize the objective function F defined by

$$F(x_i) = \sum_{i \in I} f_i x_i + \sum_{j \in J} \sum_{i \in I} c_{ij} x_{ij}^*(x_i)$$

where $x_{ij}^*(x_i)$ is an optimal solution for the user problem:

$$\min_{x_{ij}} \sum_{j \in J} \sum_{i \in I} g_{ij} x_{ij}$$

$$\text{subject to} \quad \sum_{i \in I} x_{ij} = 1, \quad j \in J,$$

$$x_{ij} \leq x_i, \qquad i \in I, j \in J,$$

$$x_{ij} \in \{0,1\}, \quad i \in I, j \in J.$$

The objective function of the decision maker is, as before, the total cost of opening facilities and servicing all users. However, now the feasible domain is described by constraints $x_i \in \{0,1\}$ for all $i \in I$ and the auxiliary optimization problem (the user problem). The values of variables x_i are known for the auxiliary problem. The bilevel problem is a new type of optimization problem. Such problems can be NP-hard even with continuous variables, linear constraints and linear objective functions [34].

The uncapacitated facility location problem with user preferences (UFLPUP) can be reduced to a single level problem [35,36]. Observe that only the ranking of the g_{ij}'s for each j is of importance and not their numerical values. Let the ranking for user j be

$$g_{i_1 j} < g_{i_2 j} < ... < g_{i_m j}.$$

Put $S_{ij} = \{l \in I |\ g_{lj} < g_{ij}\}$ for all $i \in I$. For an optimal solution $x^*_{ij}(x_i)$ of the user problem we have $x^*_{ij} = 1 \Rightarrow x_l = 0$ for all $l \in S_{ij}$. We may therefore rewrite the UFLPUP as follows:

$$\min \sum_{i \in I} f_i x_i + \sum_{j \in J} \sum_{i \in I} c_{ij} x_{ij}$$

$$\text{subject to} \quad x_{ij} + x_l \leq 1, \quad l \in S_{ij}, i \in I, j \in J,$$

$$\sum_{i \in I} x_{ij} = 1, \quad j \in J,$$

$$0 \leq x_{ij} \leq x_i, \qquad i \in I, j \in J,$$

$$x_i, x_{ij} \in \{0,1\}, \quad i \in I, j \in J.$$

Indeed, in every optimal solution of the problem all constraints of UFLP will be satisfied and the first constraint will ensure that x_{ij} is an optimal solution for the user problem. The number of variables in the problem is the same as in the UFLP. However, while the UFLP already has the large number of constraints $n+nm$, the UFLPUP has $O(m^2 n)$ additional ones. This prohibits a direct resolution except

in small instances. In order to avoid additional constraints from becoming too numerous we can rewrite them in the equivalent form:

$$\sum_{l \in S_{ij}} x_l \leq |S_{ij}|(1 - x_{ij}), \quad i \in I, j \in J,$$

or

$$x_i \leq x_{ij} + \sum_{l \in S_{ij}} x_l, \quad i \in I, j \in J,$$

or

$$x_i \leq x_{ij} + \sum_{l \in S_{ij}} x_{lj}, \quad i \in I, j \in J.$$

It is not difficult to show that the last inequality produces a better linear programming relaxation than the three previous ones [36].

The special case of the UFLPUP when $f_i = 0$ for all $i \in I$ is also interesting. For the UFLP this case is trivial; the optimal solution can be computed in linear time. However, for the UFLPUP this case is NP–hard and the integrality gap can be arbitrarily close to 1. If $c_{ij} = g_{ij}$ then we get the UFLP. If $c_{ij} = -g_{ij}$ then we can solve the problem in polynomial time [37]. Other reformulations, valid inequalities, branch and cut methods and computational results for local search methods can be found in [38,39,40].

As with the previous location problems, the UFLPUP can be reduced to the minimization problem for the pseudo–Boolean functions. For each $j \in J$ we put

$$\nabla c_{i_1 j} = c_{i_1 j}$$

$$\nabla c_{i_l j} = c_{i_l j} - c_{i_{l-1} j}, \quad 1 < l \leq m,$$

and define the pseudo–Boolean function $B(z)$ in the following way:

$$B(z) = \sum_{i \in I} f_i(1 - z_i) + \sum_{j \in J} \sum_{i \in I} \nabla c_{ij} \prod_{l \in S_{ij}} z_l.$$

Theorem 5 [35]. *The UFLPUP is equivalent to the minimization problem for the pseudo–Boolean function $B(z)$ for $z \neq (1, \ldots, 1)$. For the optimal solutions z^*, x^* of these problems we have $B(z^*) = F(x^*)$ and $z_i^* = 1 - x_i^*$ for all $i \in I$.*

Note that the coefficients ∇c_{ij} can be positive or negative. In other words, for an arbitrary pseudo–Boolean function we can reconstruct an equivalent instance of the UFLPUP and vice versa. Moreover, we can reconstruct an instance of the UFLPUP with a minimal number of users in polynomial time by the same method as in the proof of Theorem 2.

1.4. Competitive location with foresight

Let us assume that two firms want to open facilities. The first firm, which we will refer to as the leader, opens its own set of facilities X from the set I. We assume that $|X| = p$. Later, the second firm, which we will refer to as the follower, opens its own set of facilities Y from the set $I \setminus X$. We assume that $|Y| = r$. Each user selects one facility from the union $X \cup Y$ according to its own preferences, for example, according to distances to the facilities. Each firm will get a positive profit w_j if it services the user j. The firms try to maximize own profits. They do not have the same rights. The leader makes a decision first. The follower makes a decision by analyzing the set X. It is a Stakelberg game for two players, where we need to maximize the total profit of the leader [41,42,43].

Let us introduce the decision variables:

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

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

$$z_j = \begin{cases} 1 & \text{if user } j \text{ is serviced by a leader facility,} \\ 0 & \text{if user } j \text{ is serviced by a follower facility.} \end{cases}$$

For each vector x and each user j we can define the set of facilities

$$I_j(x) = \{i \in I |\, g_{ij} < \min_{l \in I}(g_{lj} |\, x_l = 1)\},$$

which allow "capturing" user j by the follower. Note that we consider conservative users [43]. If a user has the same distances to the closest leader and the closest follower facilities, he prefers the leader facility. Now the model can be written as a linear 0-1 bilevel programming problem [44]:

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

$$\text{subject to} \quad \sum_{i \in I} x_i = p,$$

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

where $z_j^*(x), y_i^*(x)$ is the optimal solution of the follower problem:

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

$$\text{subject to} \quad 1 - z_j \leq \sum_{i \in I_j(x)} y_i, \quad j \in J,$$

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

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

The objective function of the upper level defines the total profit of the leader. The feasible domain is described by two constraints and an auxiliary optimization problem of the follower. The vector x and the sets $I_j(x)$ for all $j \in J$ are known in the follower problem. The objective function of the lower level defines the total profit of the follower. The first constraint guarantees that user j is serviced by the leader if the follower has no facilities in the set $I_j(x)$. The second constraint requires opening exactly r facilities for the follower.

In [43] the follower problem is called the *medianoid* problem. It is shown there that the medianoid problem is NP–hard and that the original bilevel problem is NP–hard even for $r = 1$. Note that $z_j^*(x) = \prod_{i \in I_j(x)}(1 - y_i^*(x))$ for all $j \in J$. Hence, the follower problem can be rewritten for the polynomial $P(y, x) = \sum_{j \in J} w_j(1 - \prod_{i \in I_j(x)}(1 - y_i))$ as follows:

$$\max_y \left\{ P(y, x) \mid \sum_{i \in I} y_i = r, \quad y_i \in \{0, 1\} \text{ for all } i \in I \right\}.$$

Recall that each user is serviced by the leader or by the follower. Thus, the sum of the objective functions for the upper and lower levels is a constant. Hence, we can present the bilevel problem as a min-max problem as follows:

$$\min_x \max_y \left\{ P(y, x) \mid \sum_{i \in I} y_i = r, \sum_{i \in I} x_i = p, \; x_i, y_i \in \{0, 1\} \text{ for all } i \in I \right\}.$$

In [45] it is shown that the problem is $\sum_2^P$-hard. Therefore, we are dealing with a more difficult problem than the NP–complete problems. Polynomially solvable cases and complexity results can be found in [46]. In order to get an upper bound for the total profit of the leader we can rewrite the bilevel problem as a single level mixed integer linear program with an exponential number of constraints and variables. A similar approach is suggested in [42] for a partial enumeration algorithm. If we extract a subfamily of constraints and variables, then we may get an upper bound. In [44] a nonclassical column generation method is applied to find an optimal solution for the bilevel problem. Computational experiments for the test instances from the benchmark library *Discrete Location Problems* (http://math.nsc.ru/AP/benchmarks/english.html) indicate that the exact method allows us to find the global optimum for $p = r = 5$, $n = m = 100$.

For higher dimensions we may apply heuristics or metaheuristics. The simplest heuristic for the leader is to ignore the follower. The leader opens its own facilities to minimize the total distance between users and his facilities. He wishes to service all users and solves the classical p-median problem. This strategy is not so bad despite ignoring the follower. Computational experiments show that this lower bound can be improved by a few percent only.

The second strategy is more sophisticated. The leader anticipates that the follower will react to his decision. Therefore, $(p + r)$ facilities will be opened. According to the second heuristic, the leader solves the $(p + r)$-median problem and opens the p most profitable facilities. Unfortunately, this strategy is weak.

There is a third strategy suggested for continuous locations [47]. This heuristic is iterative. For a solution of one decision maker, we find the optimal solution for the other one. In discrete case this strategy produces a cycle. The best solution in the cycle is the result of the approach. If we use the previous strategies to create a starting solution, we can improve the profit of the leader. Surely, this is a more time consuming procedure.

One of the most powerful approaches is a hybrid memetic algorithm, where a tabu search is used to improve the elements of the population [48]. To evaluate neighboring solutions for the leader, the linear programming relaxation of the follower problem is solved by CPLEX software. To reduce the running time at each step of the tabu search, the idea of randomized neighborhoods is used. Other heuristics can be found in [49,50].

2. Local optima and Karush–Kuhn–Tucker conditions

Let us consider the minimization problem for the pseudo–Boolean function $B(z) = \sum\limits_{l \in L} \gamma_l \prod\limits_{i \in I_l} z_i$ with arbitrary coefficients γ_l. The UFLP, MSUFLP, and UFLPUP can be reduced to this problem. We wish to show the relationship between *Flip*-minimal solutions for $B(z)$ and solutions that satisfy the classical Karush–Kuhn–Tucker conditions for the Lagrange function L defined by

$$L(z, \sigma, \mu) = B(z) + \sum_{i \in I} \sigma_i (z_i - 1) - \sum_{i \in I} \mu_i z_i$$

for continuous variables $z_i \in [0, 1]$ and nonnegative multipliers σ_i, μ_i corresponding to constraints $z_i - 1 \leq 0$ and $z_i \geq 0$ for each $i \in I$. Recall that the *Flip* neighborhood for solution z, or $Flip(z)$ for short, is the set of 0-1 solutions that can be obtained from z by flipping exactly one variable. A solution is called local minimal, for example *Flip*-minimal, if it does not have neighboring solution with smaller value of the objective function.

Theorem 6. [51] *A 0-1 vector z^* is Flip-minimal if and only if there exist nonnegative multipliers σ_i^*, μ_i^*, for all $i \in I$ such that the vector (z^*, σ^*, μ^*) satisfies the Karush–Kuhn–Tucker conditions:*

(i) $\dfrac{\partial L}{\partial z_i}(z^*, \sigma^*, \mu^*) = \sum\limits_{l \in L | i \in I_l} \gamma_l \prod\limits_{j \in I_l \setminus \{i\}} z_j^* - \mu_i^* + \sigma_i^* = 0,\ i \in I;$
(ii) $z_i^* \mu_i^* = 0,\ i \in I;$
(iii) $\sigma_i^*(z_i^* - 1) = 0,\ i \in I.$

Proof. Suppose that the vector (z^*, σ^*, μ^*) satisfies the conditions (i) – (iii). Let $z' \in Flip(z^*)$ and $z_i^* = 0, z_i' = 1$ for an index $i \in I$. We then have

$$B(z^*) - B(z') = - \sum_{l \in L | i \in I_l} \gamma_l \prod_{j \in I \setminus \{i\}} z_j^* = \sigma_i^* - \mu_i^* = -\mu_i^* \leq 0.$$

Assume that $z_i^* = 1, z_i' = 0$. We have

$$B(z^*) - B(z') = \sum_{l \in L | i \in I_l} \gamma_l \prod_{j \in I \setminus \{i\}} z_j^* = \mu_i^* - \sigma_i^* = -\sigma_i^* \leq 0.$$

For both cases $B(z^*) \leq B(z')$, and z^* is *Flip*-minimal.

Consider a *Flip*-minimal vector z^*. Denote by $B_i'(z)$ the first derivative of the function $B(z)$ by the variable z_i. Put

$$\mu_i^* = \begin{cases} 0 & \text{if } z_i^* = 1 \\ B_i'(z^*) & \text{if } z_i^* = 0 \end{cases}, \qquad \sigma_i^* = \begin{cases} 0 & \text{if } z_i^* = 0 \\ -B_i'(z^*) & \text{if } z_i^* = 1 \end{cases}, \quad i \in I.$$

If $z_i^* = 0, z_i' = 1$, then $B(z^*) - B(z') = -B_i'(z^*) = -\mu_i^*$. Since $B(z^*) \leq B(z')$, we have $\mu_i^* \geq 0$. For both cases the conditions (i) – (iii) hold, which completes the proof.

Let us introduce an additional constraint $\sum_{i \in I} z_i = m - p$ into the minimization problem for $B(z)$. This new problem corresponds to the p-median problem and its generalizations. The *Swap*–neighborhood for z, or $Swap(z)$ for short, is the set of 0-1 solutions, which can be obtained from z by flipping exactly two variables with different values.

The Lagrange function L with multiplier λ and nonnegative multipliers μ_i, σ_i for each $i \in I$ is defined as follows:

$$L(z, \lambda, \mu, \sigma) = B(z) + \lambda(m - p - \sum_{i \in I} z_i) + \sum_{i \in I} \sigma_i(z_i - 1) - \sum_{i \in I} \mu_i z_i.$$

The corresponding Karush-Kuhn-Tucker conditions are presented as:

$$\frac{\partial L}{\partial z_i}(z, \lambda, \mu, \sigma) = B_i'(z) - \lambda + \sigma_i - \mu_i = 0,\ i \in I,$$

$$\sum_{i \in I} z_i = m - p,$$

$$\sigma_i(z_i - 1) = 0,\ i \in I,$$

$$\mu_i z_i = 0,\ i \in I.$$

The vector $(z^*, \lambda^*, \mu^*, \sigma^*)$ is called *a saddle point* with respect to $Swap$ neighborhood or *Swap–saddle point* if

$$L(z^*, \lambda, \mu, \sigma) \leq L(z^*, \lambda^*, \mu^*, \sigma^*) \leq L(z, \lambda^*, \mu^*, \sigma^*)$$

for all 0-1 vectors $z \in Swap(z^*)$, for all λ, and all nonnegative multipliers μ, σ.

Theorem 7. [52] *For each 0-1 vector z^* the following properties are equivalent:*
(i) z^* is Swap-minimal.
(ii) z^* satisfies the KKT conditions.
(iii) There are the multiplier λ^* and nonnegative multipliers μ_i^*, σ_i^* for each $i \in I$ such that the vector $(z^*, \lambda^*, \mu^*, \sigma^*)$ is the Swap-saddle point of the Lagrange function $L(z, \lambda, \mu, \sigma)$.

The reductions of the facility location problems to the minimization problem for the function $B(z)$ save the objective function value. Hence, the vector z is $Flip$-minimal for $B(z)$ ($Swap$-minimal) if and only if the corresponding solution S defined by $S(z) = \{i \in I \mid z_i = 0\}$ is $Flip$-minimal ($Swap$-minimal) for the location problem. When we use wider neighborhoods, for example, k-$Flip$, Lin–Kernighan neighborhoods, Fiduccia–Mattheyses neighborhoods and others [52], the set of local optima is decreased. However, all local optima satisfy the KKT conditions. Hence, the large neighborhoods extract the highest quality KKT points and we should use them in local search methods. Other theoretical properties of polynomially searchable neighborhoods can be found in [53,54,55,56].

3. Complexity of local search

3.1. The class PLS and PLS–complete problems

In order to introduce the concept of local search problems, let us recall a formal definition of an optimization problem. *An optimization problem* OP is defined by the quadruple $< \mathcal{I}, Sol, F, goal >$, where

1. $\mathcal{I}$ is the set of instances of OP;
2. Sol is a function that associates to every input instance x of OP the set of its feasible solutions;
3. F is the cost function that assigns an integer $F(s, x)$ for every feasible solution s of x;
4. $goal \in \{\min, \max\}$ specifies whether OP is a maximization or a minimization problem.

In the problem OP we need to find an optimal solution for a given instance.

Definition 1. *A local search problem* Π is a pair (OP, N), where OP is an optimization problem and N is a function that, for every pair (x, s), assigns a set $N(s, x)$ of neighboring feasible solutions. In the local search problem we need to compute a solution that does not have a better neighboring solution.

We will assume that for each instance x its feasible solutions have length bounded by a polynomial in the length of x.

Definition 2. A local search problem Π is in the class PLS if there are three polynomial-time algorithms A, B, C with the following properties:
1. For each string x, algorithm A determines whether x is an instance ($x \in \mathcal{I}$), and in this case it produces a feasible solution.
2. For each instance x and each string s, algorithm B determines whether s is a feasible solution for x and if so, B computes the cost $F(s, x)$.
3. For each instance x and each feasible solution s, algorithm C determines whether s is a local optimum, and if it is not, C outputs a neighboring solution for s with better cost.

This definition gives rise directly to the standard local search algorithm, which starts from the initial solution generated by the algorithm A, and then applies repeatedly algorithm C until it reaches a local optimum. The precise algorithm is determined by the chosen pivoting rule. For a current solution that is not a local optimum, the pivoting rule selects a neighboring solution with strictly better cost.

The class PLS is not empty. A lot of well-known combinatorial optimization problems with natural polynomial neighborhoods belong to it, for example, the traveling salesman problem with a polynomially searchable neighborhood, or the uncapacitated facility location problem with the *Flip* or *Swap* neighborhoods.

Definition 3. A local search problem Π from the class PLS belongs to the class P_{PLS} if there exists a polynomial time algorithm that returns a local optimum for every instance of the problem.

The class P_{PLS} is the polynomially solvable part of the class PLS. The relationship between the classes PLS and P_{PLS} is fundamental to complexity theory. If $\mathrm{P}_{PLS} \neq$ PLS then P$\neq$ NP.

The class P_{PLS} contains many "unweighted" problems. For example, the maximal clique problem with the *Flip*-neighborhood is in it. The number of steps of the standard local search algorithm is bounded by the number of vertices of the graph. The unweighted set covering problem with each polynomial neighborhood belongs to the class P_{PLS} as well. A nontrivial example of the local search problem from P_{PLS} is the linear programming problem with an arbitrary polynomially searchable neighborhood. It is known that the optimal solution for this problem can be found in polynomial time by the ellipsoid method. Hence, this local search problem belongs to the class P_{PLS} in spite of the fact that the simplex method is not polynomial in the worst case for many well–known pivoting rules. Note that the simplex method is in fact a local search. It moves from one basic feasible

solution to another one by exchanging a variable of the basis for another variable outside the basis.

Theorem 8 [57]. *If a PLS problem* Π *is NP–hard then NP=co–NP.*

This statement shows that it is very unlikely that the class PLS contains an NP–hard problem. Therefore, the local search problems may not be so difficult. In other words, there are no NP–complete problems that can be reduced to a local search problem from the class PLS in polynomial time. Therefore, the complexity of problems in this class is lower than that of NP–complete problems. Note that the conjecture NP$\neq$ co-NP is stronger than the conjecture P $\neq$ NP, since the coincidence of the latter two classes implies the coincidence of the former ones.

Definition 4. Let Π_1 and Π_2 be two local search problems. *A PLS–reduction* from Π_1 to Π_2 consists of two polynomial time computable functions h and g such that:
1. h maps instances x of Π_1 to instances $h(x)$ of Π_2.
2. g maps (solution of $h(x), x$) pairs to solutions of x.
3. For all instances x of Π_1, if s is a local optimum for instance $h(x)$ of Π_2, then $g(s, x)$ is a local optimum for x.

PLS–reductions have the following standard properties.

Proposition 1. *If* Π_1 *PLS–reduces to* Π_2 *and* Π_2 *PLS–reduces to* Π_3 *then* Π_1 *PLS–reduces to* Π_3. *Moreover,* $\Pi_1 \in P_{PLS}$ *if* $\Pi_2 \in P_{PLS}$.

Proposition 2. *Let* $\Pi_1 = (OP, N_1)$, $\Pi_2 = (OP, N_2)$ *be two local search problems in PLS and each local optimum under an* N_2 *neighborhood is a local optimum under an* N_1 *neighborhood. Then* Π_1 *PLS – reduces to* Π_2.

We say that a problem Π in PLS is *PLS-complete* if every problem in PLS can be PLS-reduced to it. We describe below the first PLS-complete problem, which is the basis of further reductions. This problem is called $(Circuit, Flip)$. An instance of this problem is a Boolean circuit x, which consists of AND, OR, and NOT gates. The circuit x has m inputs and n outputs. The set of feasible solutions consists of all the binary strings of length m. The neighborhood $Flip(s)$ of a solution s consists of all the binary strings of length m whose Hamming distance equals one from s. The objective function F is defined as

$$F(s) = \sum_{j=1}^{n} 2^{j-1} y_j,$$

where y_j is the j-th output of the circuit with input s.

Theorem 9 [58]. *Both the maximization version and the minimization versions of* $(Circuit, Flip)$ *are PLS-complete.*

The following local search problems are PLS-complete:

1. Graph partitioning under the following neighborhoods: KL [58], $Swap$, FM, FM_1 [57]. Given an undirected graph $G = (V, E)$ with $|V| = 2n$ and positive integer weights on its edges, we wish to find a partition of V into two subsets V_1 and V_2 with $|V_1| = |V_2| = n$, such that the sum of the weights of the edges that have one endpoint in V_1 and one endpoint in V_2 is minimal. $Swap$ neighborhood is defined as follows: a partition (V_1, V_2) has as neighbors all the partitions that can be produced by swapping a node in V_1 with a node in V_2.

In the Kernighan–Lin neighborhood we replace the single swap by a well–chosen sequence of n swaps. At each step of the sequence we choose to swap the best pair of nodes among those that have not been used in previous steps of the sequence. By the term *best*, as above, we mean that the swap produces the best improvement of the objective function. The FM neighborhood is defined in a similar way but now each step consists of the two substeps. In the first substep, we examine all the nodes that have not moved since the beginning of the sequence and choose to move the best such node from one side to the other. In the second substep, we move the best node that has not yet been moved from the opposite side. The neighborhood FM_1 contains one neighboring solution only. This solution is obtained after the first step of the FM procedure.

2. Traveling salesman problem under the k–Opt and LK' neighborhoods. Given a complete undirected graph of n nodes with positive integer weights on its edges, we wish to find the least–weight tour that passes exactly once through each node. Neighboring tours for the k–Opt neighborhood are defined as follows. We delete k edges from the tour in order to obtain k nonconnected paths. Then we reconnect these k paths so that a new tour is produced. The TSP under the k–Opt neighborhood is PLS–complete for large k [59].

The main idea of the Lin–Kernighan neighborhood is as follows. Given a tour, we delete an edge (a, b) and obtain a Hamiltonian path with end nodes a and b. Let a be stable and b variable. If we add an edge (b, c) then a circle is created. There is a unique edge (c, d) that is incident on node c, whose deletion breaks the circle, producing a new Hamiltonian path with a new variable end node d. This procedure is called rotation. We can close a tour by adding an edge between the stable end a and the variable end d. A move from the current tour to a neighboring tour consists of removing an edge, then performing a sequence of "greedy" rotations, and finally the reconnecting of the two ends to form a tour. There are many variations of this main framework depending on how exactly the rotation is chosen in each step, and on the restrictions on edges to enter and leave the tour. A variant is denoted by LK'. The PLS–completeness is shown in [60].

3. Max–Cut problem under the $Flip$ neighborhood [57]. Given an undirected graph $G = (V, E)$ with positive integer weights on its edges, we wish to find a partition of the set V into two not necessarily equal sets V_1, V_2 such that the sum of the weights of the edges that have one end point in V_1 and one end point in V_2 is maximal. The maximization and minimization versions are not equivalent. The minimization version can be solved in polynomial time, whereas the maximization

version is NP–hard. The *Flip* neighborhood is defined as in the previous case of the graph partitioning problem.

4. Max–Sat problem under the *Flip* neighborhood [61,57]. The input is a Boolean formula in a conjunctive normal form with a positive integer weight for each clause. A solution is an assignment of 0 or 1 to all variables. We wish to find an assignment such that the sum of the weights of the satisfied clauses is maximized. The restriction of Max–Sat to instances with at most k literals in each clause is called Max–kSat. In the *Flip* neighborhood two assignments are neighbors if one can be obtained from the other by flipping the value of one variable. This local search problem is PLS-complete even for $k = 2$.

5. Not–all–equal Max–Sat problem under the *Flip* neighborhood. The input is a set of clauses of the form $(\alpha_1, \ldots, \alpha_K)$, where α_i is either a literal or a constant 0 or 1. Such a clause is satisfied if its elements do not all have the same value. Each clause is assigned a positive integer weight. A set of feasible solutions of the problem consists of all assignments of 0 or 1 to the variables. We wish to find an assignment maximizing the sum of the weights of the satisfied clauses. If we restrict the clauses to have at most k literals then we get the NAE Max–kSat problem. The restriction to instances with no negative literals in their clauses is called Pos NAE Max–Sat. The *Flip* neighborhood is defined as for Max–Sat. The local search problem Pos NAE Max–3 Sat under the *Flip* neighborhood is PLS-complete [61,57].

6. Stable configurations neural networks in the Hopfield model. The input is an undirected graph $G = (V, E)$ with a weight on each edge and a threshold t_v for each node v. A configuration assigns to each node v a state $s_v \in \{-1, 1\}$. A node is "happy" if $s_v = 1$ and $\sum_u w_{(uv)} s_u s_v + t_v \geq 0$ or $s_v = -1$ and $\sum_u w_{(uv)} s_u s_v + t_v \leq 0$. A configuration is stable if all the nodes are happy. The problem is to find a stable configuration. We get an optimization problem if we introduce the cost function $\sum_{(uv)\in E} w_{(uv)} s_u s_v + \sum_{v\in V} t_v s_v$. It is known [62] that if a node is unhappy then changing its state will increase the cost. In fact, the stable configurations coincide with the locally optimal solutions with respect to the *Flip* neighborhood. This local search problem is PLS-complete [57].

Figure 1 shows the sequence of PLS–reductions for the local search problems described above and for two local search problems in facility location. The neighborhoods FM_1, FM, and KL for the $p-$median problem are defined in a way similar to the corresponding neighborhoods for the graph partitioning problem.

3.2. PLS–complete facility location problems

Theorem 10 [63]. *The local search problem (UFLP, Flip) is PLS-complete.*

Proof. Let us consider the PLS-complete problem (*Max–Cut*, *Flip*). Given a graph $G = (V, E)$ with positive weight w_e on each edge e, find a partition of

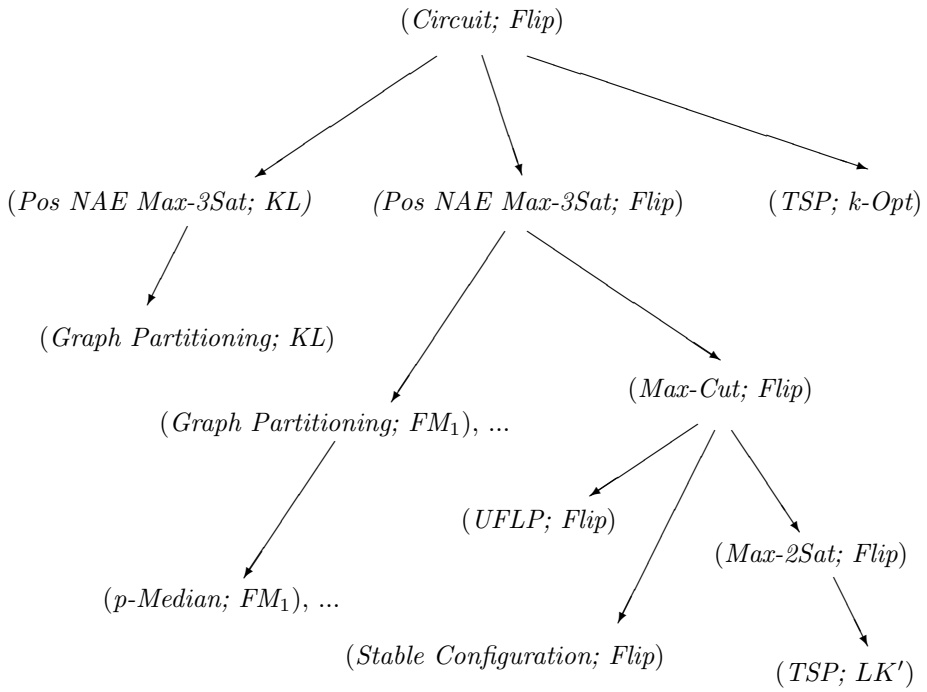

Figure 1. Reductions of the PLS–complete problems

the set V into two subsets V_1 and V_2 with maximal weight of the cut $W(V_1, V_2)$. We will reduce $(Max\text{–}Cut,\ Flip)$ to $(UFLP,\ Flip)$. To this end we present the functions h, g from Definition 4 with the required properties.

Denote by $E(i)$ the set of edges in G that are incident to the vertex $i \in V$. Put $I = V, J = E$ and

$$f_i = \sum_{e \in E(i)} w_e, \qquad c_{ie} = \begin{cases} 0 & \text{if} \quad (i = i_1) \text{ or } (i = i_2), \\ 2w_e & \text{otherwise,} \end{cases} \quad e = (i_1, i_2).$$

For each solution S of the UFLP we define a partition (V_1, V_2) in the following way: $V_1 = S,\ V_2 = V \setminus V_1$. We claim that

$$\sum_{i \in S} f_i + \sum_{j \in J} \min_{i \in S} c_{ij} + W(V_1, V_2) = 2 \sum_{e \in E} w_e.$$

This guarantees the desired properties of the reduction. Let us consider three cases for an edge $e = (i_1, i_2)$.
Case 1: $i_1, i_2 \in V_1$. The weight w_e is included in f_{i_1} and f_{i_2}, $\min\limits_{i \in S} c_{ie} = 0$, and w_e is not included in $W(V_1, V_2)$. Hence, the value w_e is presented twice in both parts of the equation.
Case 2: $i_1, i_2 \in V_2$. The values f_{i_1} and f_{i_2} are not included into the first term, $\min\limits_{i \in S} c_{ie} = 2w_e$, and w_e is not included in $W(V_1, V_2)$.

Case 3: $i_1 \in V_1, i_2 \in V_2$. The weight w_e is included in f_{i_1} and $W(V_1, V_2)$ but $\min\limits_{i \in S} c_{ie} = 0$.
Thus, we get the desired result.

Theorem 11 [52]. *The local search problem (p-median, FM_1) is PLS–complete.*

Proof. Let us consider the PLS–complete problem (*Graph Partitioning, FM_1*). Given an undirected graph $G = (V, E)$ with even number of vertices and positive weight w_e on each edge e, find a partition of the set V into two subsets V_1, V_2 with the same cardinalities and a maximal weight of the cut $W(V_1, V_2)$. We reduce this problem under the FM_1 neighborhood to (*p-median, FM_1*). Put

$$W_i = \sum_{e \in E(i)} w_e,\ W = \sum_{e \in E} w_e,\ I = \{1, \ldots, |V|\},\ J = \{1, \ldots, |E| + |V|\},\ p = |V|/2.$$

For each index $j = 1, \ldots, |E|$ we assign the edge $e \in E$ and put

$$c_{ij} = \begin{cases} 0 & \text{if } \ (i = i_1) \text{ or } (i = i_2), \\ 2w_e & \text{otherwise,} \end{cases} \qquad e = (i_1, i_2).$$

For each $j = |E| + 1, \ldots, |E| + |V|$ we define

$$c_{ij} = \begin{cases} 0 & \text{if } \ i = j - |E|, \\ W - W_i & \text{otherwise.} \end{cases}$$

For the partition (V_1, V_2) we put $S = V_1$. The proof of the theorem is based on the following equality:

$$\sum_{j \in J} \min_{i \in S} c_{ij} + W(V_1, V_2) = pW.$$

By definition we have

$$\sum_{j=1}^{|E|} \min_{i \in S} c_{ij} = 2 \sum (w_e \mid e = (i_1, i_2), i_1, i_2 \notin S)$$

and

$$\sum_{j=1+|E|}^{|J|} \min_{i \in S} c_{ij} = \sum_{i \notin S} (W - W_i) = pW - \sum_{i \notin S} W_i.$$

Note that

$$\sum_{i \notin S} W_i = W(V_1, V_2) + \sum_{j=1}^{|E|} \min_{i \in S} c_{ij},$$

which completes the proof.

3.3. Complexity of the standard local search algorithm

As we have mentioned above, there are polynomially solvable local search problems in the class PLS. However, we still do not know of polynomial time algorithms for PLS–complete problems. Whether such algorithms exist or not is still an open question for further research. Below we will observe that the standard local search algorithm is not appropriate since it takes, in the worst case, an exponential number of iterations to reach a local optimum for every pivoting rule. In other words, we need a fresh idea (new framework, new "ellipsoid" method) to show that PLS=P$_{PLS}$, if it is true.

Definition 5. Let Π be a local search problem and x be an instance of Π. *The transition graph* $TG_\Pi(x)$ of the instance x is a directed graph with one node for each feasible solution of x and with an arc $(s \to t)$ whenever $t \in N(s,x)$ and $F(t,x) < F(s,x)$. *The height of a node* v is the length of the shortest path in $TG_\Pi(x)$ from v to a sink (a vertex with no outgoing arcs). *The height of* $TG_\Pi(x)$ is the largest height of a node.

The height of a node v is a lower bound on the number of iterations needed by the standard local search algorithm even if it uses the best possible pivoting rule.

Definition 6. Suppose Π_1 and Π_2 are problems in PLS and let (h,g) be a PLS–reduction from Π_1 to Π_2. This reduction is *tight* if for every instance x of Π_1 we can choose a subset R of feasible solutions for the image instance $h(x)$ so that the following properties are satisfied:
1. R contains all local optima of $h(x)$.
2. For every solution t of x we can construct in polynomial time a solution $q \in R$ of $h(x)$ such that $g(q,x) = t$.
3. Suppose that the transition graph of $h(x)$, $TG_{\Pi_2}(h(x))$, contains an arc from $q \in R$ to $q' \in R$ or a directed path from q to q' such that all internal path nodes are outside R, and let $t = g(q,x)$ and $t' = g(q',x)$ be the corresponding solutions of x. Then either $t = t'$ or $TG_{\Pi_1}(x)$ contains an arc from t to t'.

Tight reductions allow us to transfer lower bounds on the running time of the standard local search algorithm from one problem to another. Thus, if the standard local search algorithm of Π_1 takes exponential time in the worst case, then so does the standard algorithm for Π_2.

All PLS–complete problems that we have referred to are complete under tight PLS reductions. We now want to show that in the worst case the running time of the standard local search algorithm is exponential for the tightly PLS-complete problems. To prove this it suffices to find a local search problem in the class PLS which has this property.

Lemma 2 [64]. *There is a local search problem in PLS whose standard local search algorithm takes exponential time.*

Proof. Consider the following artificial minimization problem. For every instance x of size n, the solution set consists of all n-bit integers $0, \ldots, 2^n - 1$. For each solution i, its cost is i, and if $i > 0$ it has one neighbor, $i - 1$. Thus, there is a unique local and global minimum, namely 0, and the transition graph is a path from $2^n - 1$ down to 0. The local search algorithm starting at $2^n - 1$ will follow this path.

Theorem 12. *The standard local search algorithm takes exponential time in the worst case for the following problems, regardless of the tie-breaking and pivoting rules used:*

- *Graph partitioning under the neighborhoods KL [58], $Swap$, FM, FM_1 [57].*
- *Traveling salesman problem under the k-Opt neighborhood for some constant k [59], and the LK' neighborhood [60].*
- *Max-Cut, Max-2Sat and $Pos\ NAE\ Max$-3Sat under the $Flip$ neighborhood [61,57].*
- *Stable configuration for neural networks [57].*
- *Uncapacitated facility location problem under the $Flip$ neighborhood [63].*
- *p-median problem under the FM_1, $Swap$, KL, FM neighborhoods [65].*

It is interesting to see a family of nonartificial instances and initial solutions for which the local search algorithm takes an exponential number of iterations to find a local optimum. In order to do so we consider the Generalized Graph 2-Coloring problem (2-$GGCP$) with the $Flip$-neighborhood: given an undirected graph $G = (V, E)$ with integer weight w_e on each edge e, find a color assignment $c : V \to \{1, 2\}$ of the vertices that minimizes the total weight of the monochromatic edges. For each solution $c(V)$, a $Flip$ neighbor is obtained by choosing a vertex and assigning a new color. A solution is $Flip$-optimal if the flipping of every single vertex does not decrease the total weight of monochromatic edges. The local search problem (2$GGCP$, $Flip$) is tightly PLS-complete as a reformulation of the Max-Cut problem under the $Flip$ neighborhood.

To illustrate the exponential number of iterations needed for finding a $Flip$-optimal solution, we present an example of a graph and an initial solution for 2$GGCP$ for which the *best improvement*, meaning always flipping the best vertex, takes an exponential number of iterations. This graph consists of K modules with weights on the edges as shown in Figure 2 for $i = 1, \ldots, K$ and a chain of three additional vertices as shown in Figure 3. Vertex 1 is called the input node and vertex 7 is called the output node of a module. The input node of module i is adjacent to the output node of module $i + 1$, for $i = K - 1, \ldots, 1$, and the input node of module K is adjacent to the rightmost vertex of the chain of Figure 3. The output node of module 1 is only adjacent to vertices 4, 5, 6, and 10 of this module. An edge of weight $-M$, where M is some large positive value, makes sure that the two vertices incident to this edge have the same color.

Let us consider a starting solution with vertices of the same color. In this case, only flipping the rightmost vertex of the chain yields an improvement. This flip results in a solution in which the input node of module K is "unhappy".

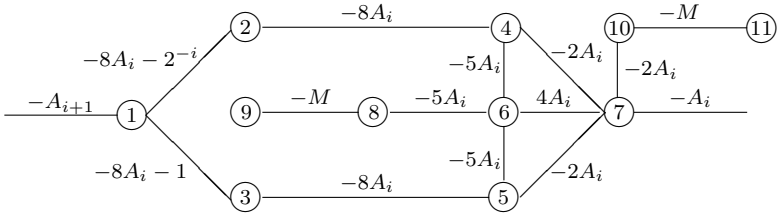

Figure 2. Module $i : A_i = 20^{i-1}$

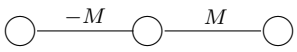

Figure 3. Chain

Theorem 13 [66]. *If the input node of module K is the only unhappy node, then the output node of module 1 flips 2^K times.*

3.4. Average case behavior

The worst–case analysis yields bad predictions about local search and metaheuristics such as Genetic algorithms, Variable Neighborhood Search, Ant Colony algorithms, and others [67,68,69]. However, computational experiments show a lot of positive results for local search. In practice, it finds a local optimum quickly, with a small number of iterations as compared to the size of the search space. It is interesting to understand why the local search methods are so successfully used in applications. Is the standard local search algorithm polynomial on average? The first results in this direction have been obtained by C. Tovey [70,71].

Consider the problem of maximizing a real valued function F defined on the vertices of the n–dimensional 0–1 hypercube. We assume that the values of F are distinct. In the *Flip* neighborhood for the hypercube, two vertices are neighbors if they differ in exactly one component. For each F we can construct an *ordering*: a list of the vertices from best to worst function value. The random distribution we consider is such that all orderings are equally likely to occur.

Theorem 14 [72]. *Under the assumption that all orderings are equally likely, the expected number of iterations of the standard local search algorithm with the Flip neighborhood and every pivoting rule is less than $\frac{3}{2}en$, where e is the logarithmic constant.*

Note, that the statement holds for every kind of pivoting rule. Even a careful rule such as picking the worst better neighbor has an expected performance of less than $\frac{3}{2}en$.

Theorem 15 [72]. *Suppose the ratio of probabilities of occurrence satisfies*

$$\frac{Prob\ [v]}{Prob\ [v']} \leq 2^{\alpha n}$$

for all orderings v, v' *and a positive constant* α*. Then the expected number of iterations of every local search algorithm with the Flip neighborhood is less than* $(\alpha + 2)en$*.*

This statement can be extended for more powerful neighborhoods.

Theorem 16 [72]. *Suppose the vertices of the hypercube are assigned neighbors in such a way that every vertex has at most* $q(n)$ *neighbors, where* $q(n) \geq n$ *is a polynomial. Then for every probability distribution satisfying*

$$\frac{Prob\ [v]}{Prob\ [v']} \leq 2^{\alpha n} \quad \text{for all orderings } v, v',$$

the expected number of iterations of every local search algorithm is less than $e(\alpha + 2)q(n)$*.*

Other results for more powerful neighborhoods and other random distributions can be found in [72]. It is not clear how to use these results for the facility location models. Can we get a polynomial upper bound for the running time of the standard local search algorithm on average for (UFLP, *Flip*), (p-median, *Swap*), and others? This is an open question for further research.

3.5. Approximate local search

Computational studies of local search algorithms and metaheuristics have been extensively reported in the literature for various NP–hard optimization problems. Empirically, local search heuristics appear to converge very quickly, in low–order polynomial time. The standard local search algorithm terminates in a pseudo polynomial number of iterations, but polynomial–time algorithms for finding a local optimum are not known in general. It is interesting to explore the possibility of identifying *approximately locally optimal* solutions in polynomial time. We say that a feasible solution s^ε to an instance of a combinatorial optimization problem OP with neighborhood function N is an *ε-local minimum* if

$$F(s^\varepsilon) \leq (1 + \varepsilon)F(s) \quad \text{for all } s \in N(s^\varepsilon) \text{ and some positive } \varepsilon.$$

Hence, while s^ε is not necessarily a local minimum, it *almost* is. A family of algorithms $(A_\varepsilon)_{\varepsilon>0}$ for the local search problem is an *ε-local optimization scheme* if A_ε produces an ε-local optimum. If the running time of algorithm A_ε is polynomial in the input size and $1/\varepsilon$, it is called a *fully polynomial-time ε-local optimization scheme.* In [73] it is shown that every local search problem in the class PLS with a linear objective function has a fully polynomial-time ε-local optimization scheme.

In particular, an ε-locally optimal solution can be computed in polynomial time for the PLS–complete problems mentioned above.

Let us consider a linear combinatorial optimization problem OP, where the set Sol of feasible solutions is a family of subsets of a finite ground set $E = \{1, \ldots, n\}$. The objective function $F : 2^E \to Q_+$ assigns a nonnegative cost to every feasible solution $s \in Sol$ through $F(s) = \sum_{e \in s} f_e$. Note that we focus on a linear combinatorial optimization problem as opposed to a *general* combinatorial optimization problem. The class of problems we are looking at is equivalent to that of 0–1 integer linear programming problems. The UFLP, MSUFLP and UFLPUP belong to this class.

The algorithm starts with a feasible solution s^0. We then alter the element costs f_e for $e \in E$ according to a prescribed scaling rule to generate a modified instance. Using local search on this modified problem, we look for a solution with an objective function value (with respect to the original cost) that is half that of $F(s^0)$. If no solution is found then we are at a local optimum for the modified problem and output this solution. Otherwise, we replace s^0 by the solution, which has a cost of at most $0,5F(s^0)$, and the algorithm is repeated. All details of the algorithm are described below. Note that the modification of the cost coefficients in Step 2 merely amounts to rounding them up to the closest integer multiple of q.

Algorithm ε-Local Search

1 Find $s^0 \in Sol(x)$ and put $i := 0$.

2 Put $K := F(s^i)$, $q := \varepsilon K/(2n(1+\varepsilon))$, $f'_e := \lceil \frac{f_e}{q} \rceil q$, $e \in E$.

3 Put $j := 0$ and $s^{ij} := s^i$.

4 Repeat

If s^{ij} is a local minimum then $s^\varepsilon := s^{ij}$, STOP

else select better solution $s^{ij+1} \in N(s^{ij})$, $F(s^{ij+1}) < F(s^{ij})$ and put $j := j + 1$.

until $F(s^{ij}) \leq K/2$

5 Put $s^{i+1} := s^{ij}$, $i := i + 1$ and goto 2.

Theorem 17 [73]. *Algorithm ε-Local Search produces an ε-local minimum and its running time is polynomial in the input size and $1/\varepsilon$.*

Proof. Let s^ε be the solution produced by the algorithm and $s \in N(s^\varepsilon)$. Note that

$$F(s^\varepsilon) = \sum_{e \in s^\varepsilon} f_e \leq \sum_{e \in s^\varepsilon} \lceil \frac{f_e}{q} \rceil q \leq \sum_{e \in s} \lceil \frac{f_e}{q} \rceil q \leq \sum_{e \in s} q(\frac{f_e}{q} + 1) \leq \sum_{e \in s} f_e + nq = F(s) + nq.$$

If $F(s) \geq K/2$ then

$$\frac{F(s^\varepsilon) - F(s)}{F(s)} \leq \frac{nq}{F(s)} \leq \frac{nq}{F(s^\varepsilon) - nq} \leq \frac{2nq}{K - 2nq} = \varepsilon.$$

Let us analyze the running time. Step 1 is polynomial because the local search problem is in the class PLS. In each improvement move in Step 4 we get an improvement of at least of q units. Thus the number of local search iterations in Step 4 is $O(n(1+\varepsilon)/\varepsilon) = O(n/\varepsilon)$. Step 2 is executed at most $\log F(s^0)$ times. Thus, the total number of local search iterations is $O(n \log F(s^0)/\varepsilon)$. More accurate calculations give $O(n^2\varepsilon^{-1} \log n)$ iterations.

If we replace the relative error in the definition of an ε-local optimum with the absolute error, then the existence of a polynomial ε–Local Search algorithm will imply PLS=P$_{PLS}$.

Theorem 18 [73]. *If there is an algorithm that for every instance x of a PLS-complete local search problem (OP, N) finds a feasible solution s^ε in polynomial time, such that*

$$F(s^\varepsilon) \leq F(s) + \varepsilon \text{ for all } s \in N(s^\varepsilon)$$

for some fixed positive ε, then PLS=P_{PLS}.

Proof. Recall that the objective function is an integer–value. For each instance x we create a new instance x' with the same set of feasible solutions, $Sol(x') = Sol(x)$, and a new objective function defined by

$$F'(s) = \sum_{e \in s} f'_e, \text{ for all } s \in Sol(x'), \text{ where } f'_e = f_e(1+\varepsilon) \text{ for every } e \in E.$$

We apply the algorithm to the new instance x' and let s' be the resulting solution. Then, $F'(s') - F'(s) \leq \varepsilon$ for all $s \in N(s')$. Thus, $F(s') - F(s) \leq \varepsilon/(\varepsilon+1) < 1$ for all $s \in N(s')$ and s' is a local optimum for x.

Theorem 19 [73]. *If a PLS-complete local search problem (OP, N) has a fully polynomial time ε-local optimization scheme $(A_\varepsilon)_{\varepsilon>0}$ such that the actual running time of A_ε is polynomial in the input size and $\log 1/\varepsilon$, then PLS=P_{PLS}.*

Proof. Choose $\varepsilon = 1/(nf_{\max}+1)$, $f_{\max} = \max\limits_{e \in E} f_e$ and apply A_ε. Note that its running time is polynomial in the input size. If s^ε is the solution returned by the algorithm, then $F(s^\varepsilon) \leq (1+\varepsilon)F(s) < F(s) + 1$ for all $s \in N(s^\varepsilon)$. Hence, s^ε is a local optimum.

Observe that the results hold for the facility location problems as well.

4. The quality of local optima

We say that a neighborhood is exact if each local optimum is a global one. The standard local search algorithm with an exact neighborhood produces the optimal solution of the problem. However, for the p-median problem (the traveling salesman problem [74] and some others) the existence of the polynomially search-

able exact neighborhoods implies P=NP [65]. In general, this property can be presented in the following way.

An optimization problem OP with optimal value $F^*(x)$ is called *pseudo polynomially bounded* if there is a polynomial q in the length of $x \in \mathcal{I}$ and $Max(x)$, which is the maximal absolute value of the components of x, such that

$$|F(s) - F^*(x)| \leq q(|x|, Max(x))$$

for all $x \in \mathcal{I}$ and for all $s \in Sol(x)$. The set of the pseudo polynomially bounded problems is denoted by NPO_B.

Theorem 20 [65,64]. *Let $OP \in NPO_B$ and (OP, N) $\in$ PLS. If the approximation of OP within a factor ε is strongly NP–hard then N cannot guarantee a ratio of ε unless P=NP.*

For $\varepsilon = 1$ we have the following.

Corollary 1. *If $OP \in NPO_B$, (OP, N) $\in$ PLS and OP is strongly NP–hard then N cannot be exact unless P=NP.*

Theorem 21 [64]. *If $(OP, N) \in PLS$ and the approximation of OP within a factor ε is NP–hard then N cannot guarantee a ratio of ε unless NP=co–NP.*

Corollary 2. *If $(OP, N) \in PLS$ and OP is NP–hard, then N cannot be exact unless NP=co–NP.*

As we have mentioned in Section 1, the uncapacitated facility location problem is NP–hard in the strong sense even for the metric case. Moreover, the existence of a polynomial time 1.463–factor approximation algorithm for this problem implies P=NP. Therefore, it is difficult to find an exact polynomially searchable neighborhood for this problem or a neighborhood that guarantees the ratio $\varepsilon \leq 1.463$. This is bad news. However, below we will present some good news.

We say that an optimization problem OP is *polynomially bounded* if there exists a polynomial r such that $F(s) \leq r(|x|)$ for every instance x of OP and every feasible solution s of x.

Definition 7. An optimization problem OP has *guaranteed local optima* if there exists a polynomial time searchable neighborhood N and a constant k such that $F(s) \leq kF^*(x)$ for every instance x of OP and every local minimum s of x with respect to N.

Definition 8. For an instance x and feasible solutions s, s' of x we say that s' is an *h-bounded neighbor of s* if the Hamming distance between s and s' is at most h. A neighborhood N is said to be *h-bounded* if there exists a constant h such that every neighbor of s is an h-bounded for every feasible solution s and every instance x.

Definition 9. Let OP be a polynomially bounded optimization problem. We say that OP *belongs to the class Guaranteed Local Optima* (GLO) if the following two conditions hold:

– at least one feasible solution s of x can be computed in polynomial time for every instance x of OP,
– there exists a constant h such that OP has a guaranteed local optima with respect to a suitable h-bounded neighborhood.

Definition 10. Let A and B be two optimization problems. A is said to be *PTAS-reducible* to B (in symbol $A \leq_{PTAS} B$) if three functions f, g, c exist such that:

– for every $x \in \mathcal{I}_A$ and for every $\varepsilon \in (0,1)_Q$, (Q is the set of rational numbers) $f(x,\varepsilon) \in \mathcal{I}_B$ is computable in polynomial time with respect to $\mid x \mid$;
– for every $x \in \mathcal{I}_A$, for every $s \in Sol_B(f(x,\varepsilon))$, and for every $\varepsilon \in (0,1)_Q$, $g(x,s,\varepsilon) \in Sol_A(x)$ is computable in time polynomial with respect to both $|x|$ and $|s|$;
– $c : (0,1)_Q \to (0,1)_Q$ is computable and surjective;
– for every $x \in \mathcal{I}_A$, for every $s \in Sol_B(f(x,\varepsilon))$, and for every $\varepsilon \in (0,1)_Q$ $E_B(f(x,\varepsilon),s) \leq c(\varepsilon)$ implies $E_A(x,g(x,s,\varepsilon)) \leq \varepsilon$, where $E(x,s)$ is the relative error of s for x,

$$E(x,s) = \frac{|F(s) - F^*(x)|}{\max\{F(s), F^*(x)\}}.$$

Suppose that $A \leq_{PTAS} B$ and $B \in APX$, then $A \in APX$. If C is a class of optimization problems, then by $\overline{C}$ we denote the *closure* of C under PTAS reductions, that is, the set of problems defined by

$$\overline{C} = \{A \mid \exists B \in C \text{ such that } A \leq_{PTAS} B\}.$$

Theorem 22 [75]. $\overline{\text{GLO}} = \text{APX}$.

In other words, *"... the basis of approximability of a large class problems stands an important combinatorial property, namely, the fact that all local optima have guaranteed quality with respect to global optima"* G. Ausiello, M. Protasi [75].

5. Computationally difficult instances

The iterative local search methods show high performance for many combinatorial optimization problems in business, engineering, and science. These metaheuristics provide fast and robust tools, producing high quality solutions for location problems as well [77,78,79,76]. As a rule, they deal with the set of local optima under polynomially searchable neighborhoods. If the local optima cluster is in a small part of the feasible domain, as for the metric TSP [80], we understand why these heuristics are so effective. Conversely, if we wish to generate computation-

ally difficult instances for the local search methods then we may try to create instances, where local optima are scattered all around the feasible domain. Such computationally difficult instances for the UFLP based on binary perfect codes, finite projective planes, and others [63] are described below.

5.1. POLYNOMIALLY SOLVABLE INSTANCES

Let us consider a finite projective plane of order k [81], which is a collection of $n = k^2 + k + 1$ points $x_1, \dots x_n$ and lines $L_1, \dots, L_n$. An incidence matrix A is an $n \times n$ matrix defining the following: $a_{ij} = 1$ if $x_j \in L_i$ and $a_{ij} = 0$ otherwise. The incidence matrix A satisfying the following properties:

1. A has constant row sum $k + 1$;
2. A has constant column sum $k + 1$;
3. the inner product of every pair of distinct rows of A is 1;
4. the inner product of every pair of distinct columns of A is 1.

These matrices exist if k is a power of a prime. A set of lines $B_j = \{L_i | x_j \in L_i\}$ is called a bundle for the point x_j. The cardinality of each bundle is $k + 1$ and $|B_{j1} \cap B_{j2}| = 1$ for every pair of different points x_{j1} and x_{j2}. Let us define a class of instances for the UFLP. Put $I = J = \{1, \dots, n\}$ and

$$c_{ij} = \begin{cases} \xi_{ij} & \text{if } a_{ij} = 1, \\ +\infty & \text{otherwise,} \end{cases} \qquad f_i = f \text{ for all } i \in I, \text{ where } f > \sum_{i \in I} \sum_{j \in J} \xi_{ij}.$$

We denote this class by FPP_k. It is easy to see that the optimal solution for FPP_k corresponds to a bundle. Hence, the problem can be solved in polynomial time.

Every bundle corresponds to a strong local optimum of the UFLP under the neighborhood $Flip \cup Swap$. Global optimum is one of them. The Hamming distance for an arbitrary pair of the strong local optima equals $2k$. Hence, the diameter of the area, where local optima are located, is quite large. Moreover, there are no other local optima with distance to the bundle less than or equal to k. As we will see in some computational results, the local optima have large basins of attraction. For metaheuristics it is an additional obstacle for moving from one local optimum to another. In Tabu Search we have to use a large tabu list. For Simulated Annealing we need high temperatures. If the population in Genetic Algorithm is a collection of the bundles then the crossover operators produce "bad" local optima or the same bundles. For the GRASP heuristic this class is difficult too [76].

5.2. INSTANCES WITH EXPONENTIAL NUMBER OF STRONG LOCAL OPTIMA

Let us consider two classes of instances, where the number of strong local optima grows exponentially as dimension increases. The first class uses binary perfect codes with code distance 3. The second class involves a chess board.

Let B_k be a set of words (or vectors) of length k over an alphabet {0,1}. A binary code of length k is an arbitrary nonempty subset of B_k. A binary perfect code C with distance 3 is a subset of B_k with property $|C| = 2^k/(k+1)$ such that the Hamming distance $d(c_1, c_2)$ for all $c_1, c_2 \in C$ is at least 3 whenever $c_1 \neq c_2$. These codes exist for $k = 2^r - 1$ and $r > 1$, integer. Put $n = 2^k, I = J = \{1, \ldots, n\}$. Every element $i \in I$ corresponds to a vertex $x(i)$ of the binary hypercube $\mathbf{Z}_2^k$. Therefore, we may use a distance $d_{ij} = d(x(i), x(j))$ for every pair of elements $i, j \in I$. Now we define

$$c_{ij} = \begin{cases} \xi_{ij} & \text{if } d(x(i), x(j)) \leq 1, \\ +\infty \text{ otherwise,} \end{cases} \qquad f_i = f \text{ for all } i \in I.$$

An arbitrary perfect code C produces a partition of $\mathbf{Z}_2^k$ into $2^k/(k+1)$ disjointed spheres of radius 1 and corresponds to a strong local optimum for the UFLP. The number of perfect codes $\aleph(k)$ grows exponentially as k increases. The best known lower bound [82] is

$$\aleph(k) \geq 2^{2^{\frac{k+1}{2}\log_2(k+1)}} \cdot 3^{2^{\frac{k-3}{4}}} \cdot 2^{2^{\frac{k+5}{4}\log_2(k+1)}}.$$

The minimal distance between two perfect codes or strong local minima is at least $2^{(k+1)/2}$. We denote the class of benchmarks by BPC_k.

Let us glue boundaries of the $3k \times 3k$ chess board so that we get a torus. Put $r = 3k$. Each cell of the torus has 8 neighboring cells. For example, the cell $(1, 1)$ has the following neighbors: $(1, 2), (1, r), (2, 1), (2, 2), (2, r), (r, 1), (r, 2), (r, r)$. Define $n = 9k^2$, $I = J = \{1, \ldots, n\}$ and

$$c_{ij} = \begin{cases} \xi_{ij} & \text{if the cells } i, j \text{ are neighbors,} \\ +\infty \text{ otherwise,} \end{cases} \qquad f_i = f \text{ for all } i \in I.$$

The torus is divided into k^2 squares with 9 cells in each of them. Every cover of the torus by k^2 squares corresponds to a strong local optimum for the UFLP. The total number of these strong local optima is $2 \cdot 3^{k+1} - 9$. The minimal distance between them is $2k$. We denote this class of benchmarks by CB_k.

5.3. INSTANCES WITH LARGE INTEGRALITY GAP

As we will see later, the integrality gap for the described classes is quite small. Therefore, the branch and bound algorithm finds an optimal solution and proves the optimality quickly. It is interesting to design benchmarks, which are computationally difficult for both metaheuristics and branch and bound methods.

As in previous cases, let the $n \times n$ matrix (c_{ij}) have the following property: each row and column have the same number of finite elements. We denote this number by l. The value l/n is called the density of the matrix. Now we present an algorithm to generate random matrices (c_{ij}) with the fixed density.

Random matrix generator (l, n)

1. $J \leftarrow \{1, \ldots n\}$
2. Column $[j] \leftarrow 0$ for all $j \in J$
3. $c[i, j] \leftarrow +\infty$ for all $i, j \in J$
4. **for** $i \leftarrow 1$ **to** n
5. **do** $l_0 \leftarrow 0$
6. **for** $j \leftarrow 1$ **to** n
7. **do if** $n - i + 1 = l -$ Column$[j]$
8. **then** $c[i, j] \leftarrow \xi[i, j]$
9. $l_0 \leftarrow l_0 + 1$
10. Column$[j] \leftarrow$ Column$[j] + 1$
11. $J \leftarrow J \setminus j$
12. select a subset $J' \subset J, |J'| = l - l_0$ at random and put $c[i, j] \leftarrow \xi[i, j]$, for $j \in J'$.

The array Column$[j]$ keeps the number of small elements in the j-th column of the generating matrix. Variable l_0 is used to count the columns, where small elements must be located in the i-th row. These columns are detected in advance (line 7) and removed from the set J (line 11). Note that we may get random matrices with exactly l small elements for each row only if we remove lines 6-11 from the algorithm. By transposing we get random matrices with this property for columns only. Now we introduce three classes of benchmarks:
Gap-A: each column of c_{ij} has exactly l small elements;
Gap-B: each row of c_{ij} has exactly l small elements;
Gap-C: each column and row of c_{ij} have exactly l small elements.

For these classes we save $I = J = \{1, \ldots, n\}$ and $f_i = f$ for all $i \in I$. The instances have a significant integrality gap $\delta = 100\%(F^* - F_{LP})/F^*$, where F_{LP} is an optimal value for the linear programming relaxation. For $l = 10, n = 100$ we observe that $\delta \in [21\%, 29\%]$. As a consequence, the branch and bound algorithm evaluates about $0,5 \cdot 10^9$ nodes in the branching tree for most of the instances from the class Gap-C.

5.4. COMPUTATIONAL EXPERIMENTS

To study the behavior of metaheuristics, we generate 30 random test instances for each class. The values of ξ_{ij} are taken from the set $\{0, 1, 2, 3, 4\}$ at random and $f = 3000$. All instances are available at http://www.math.nsc.ru/AP/benchmarks/english.html. Optimal solutions are found by the branch and bound algorithm. Table 1 shows the performance of the algorithm on average. Column *Running time* presents the execution times on a PC Pentium 1200 MHz, RAM 128 Mb. Column *Iterations B&B* shows the total number of iterations or number of evaluated nodes in the branching tree. Column *The best iteration* shows iterations for which optimal solutions were discovered. For comparison we include two well known classes:

Table 1. Performance of the branch and bound algorithm in average

Benchmarks classes	n	**Gap** δ	**Iterations B&B**	**The best iteration**	**Running time**
BPC_7	128	0.1	374 264	371 646	00:00:15
CB_4	144	0.1	138 674	136 236	00:00:06
FPP_{11}	133	7.5	6 656 713	6 635 295	00:05:20
$Gap-A$	100	25.6	10 105 775	3 280 342	00:04:52
$Gap-B$	100	21.1	30 202 621	14 656 960	00:12:24
$Gap-C$	100	28.4	541 320 830	323 594 521	01:42:51
$Uniform$	100	4.7	9 834	2 748	<00:00:01
$Euclidean$	100	0.1	1 084	552	<00:00:01

Table 2. Attributes of the local optima allocation

Benchmarks classes	N	Diameter	Radius			R_{100}	R^*
			min	ave	max		
BPC_7	8868	55	1	3	357	24	52
CB_4	8009	50	1	13	178	78	53
FPP_{11}	8987	51	1	2	8	3	1
$Gap-A$	6022	36	1	53	291	199	7
$Gap-B$	8131	42	1	18	164	98	16
$Gap-C$	8465	41	1	14	229	134	21
$Uniform$	1018	33	1	31	101	61	1
$Euclidean$	40	21	11	13	18	–	10

Uniform: values c_{ij} are selected in the interval $[0,\ 10^4]$ at random with uniform distribution and independently from each other.

Euclidean: values c_{ij} are Euclidean distances between points i and j in two-dimensional space. The points are selected in a square of size 7000×7000 at random with uniform distribution and independently from each other.

For these classes $f = 3000$. The interval and size of the square are taken in such a way that optimal solutions have the same cardinality as in the previous classes. Table 1 confirms that classes $Gap-A$, $Gap-B$, and $Gap-C$ have a large integrality gap and they are the most difficult for the branch and bound algorithm. The classes BPC_7, CB_4, and $Euclidean$ have a small integrality gap. Nevertheless, the classes BPC_7 and CB_4 are more difficult than the $Euclidean$ class. This has a simple explanation: classes BPC_7 and CB_4 have many strong local optima with small waste over the global optimum.

In order to understand the difference between classes from the point of view of local optima allocation, we produce the following computational experiment. For 9000 random starting points we apply the standard local improvement algorithm with the $Flip \cup Swap$ neighborhood and get a set of local optima, some of which are identical. Impressive differences between benchmark classes become clear when the cardinality of the local optima sets are compared. Classes $Uniform$ and

Euclidean have small pathological cardinalities of local optima sets and, as we will see below, these classes are very easy for metaheuristics. In Table 2 the column N shows the cardinalities for typical instances in each class. The column *Diameter* yields a lower bound for the diameter of area, where local optima are located. This value equals the maximal mutual Hamming distance over all pairs of local optima obtained.

Figures 4 – 11 plot the costs of local optima against their distances from the global optimum. For every local optimum we draw a sphere. The center of the sphere has coordinates (x, y), where x is the distance, and y is the value of the objective function. The radius of the sphere is the number of local optima, which are located near this local optimum. More precisely, the Hamming distance d is less than or equal to 10. In Table 2 columns min, ave, and max show the minimal, average, and maximal radiuses for the corresponding sets of local optima. The class FPP_{11} has an extremely small maximal and average value of the radiuses. Hence, the basins of attraction for the local optima are quite large. In Figure 6 the local optima, which correspond to the bundles, are shown by two lower spheres. The distance between them is 22. One of them is the global optimum. All other local optima are located quite far from global optimum and have higher values of the objective function. The distance from global optimum to the nearest local optimum is 12. Column R^* in Table 2 gives the radius of the sphere for the global optimum. It equals 1 for the classes FPP_{11} and $Uniform$. The maximal values 53 and 52 belong to classes CB_4 and BPC_7. Note that for all classes the sphere of global optima is not the maximal or minimal one. It seems that there is no

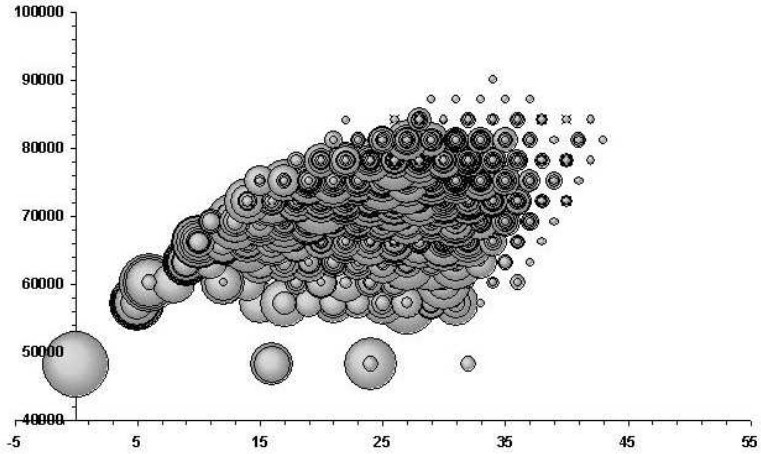

Figure 4. Analysis of local optima for the class BPC_7

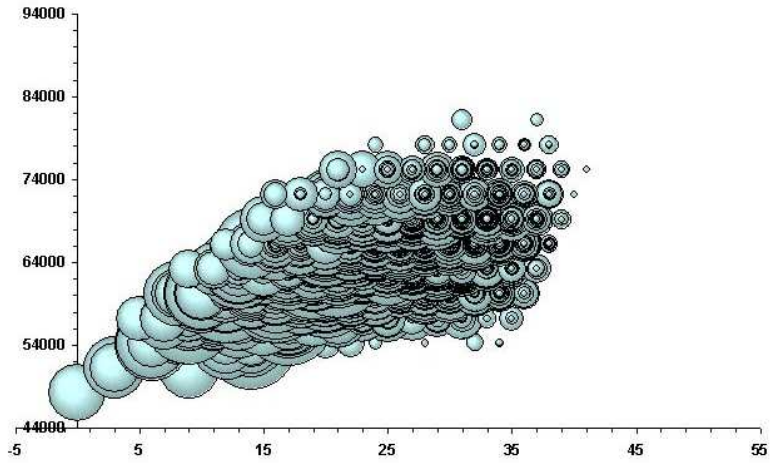

Figure 5. Analysis of local optima for the class CB_4

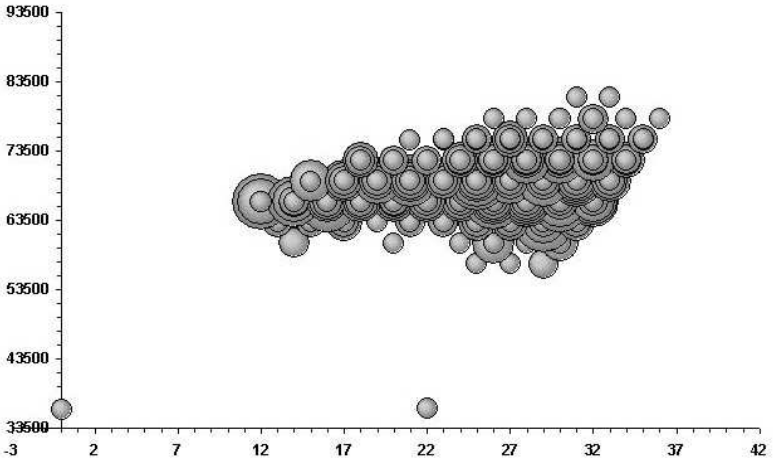

Figure 6. Analysis of local optima for the class FPP_{11}

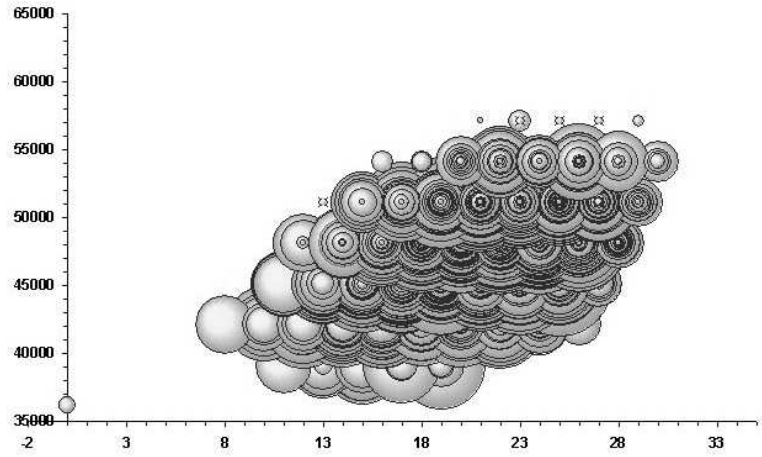

Figure 7. Analysis of local optima for the class $GAP - A$

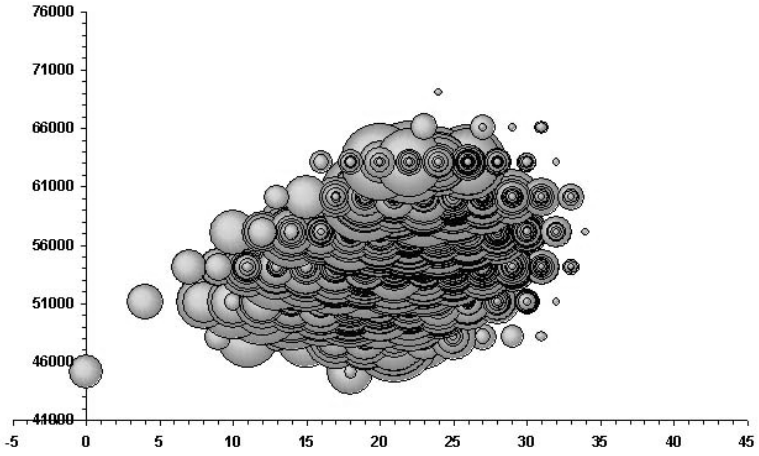

Figure 8. Analysis of local optima for the class $GAP - B$

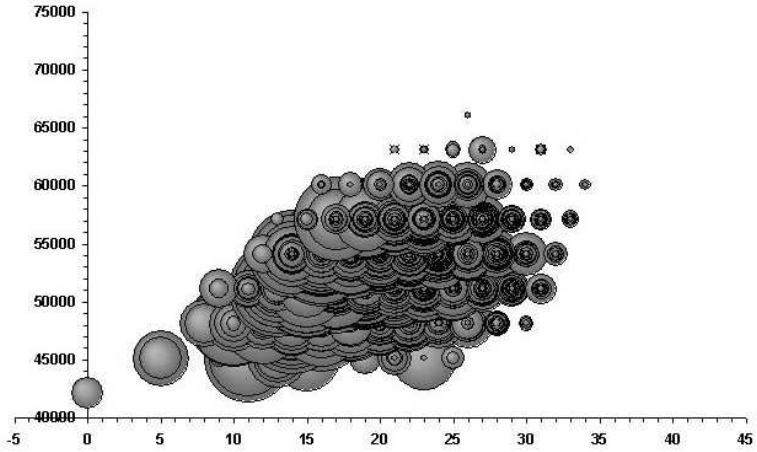

Figure 9. Analysis of local optima for the class $GAP - C$

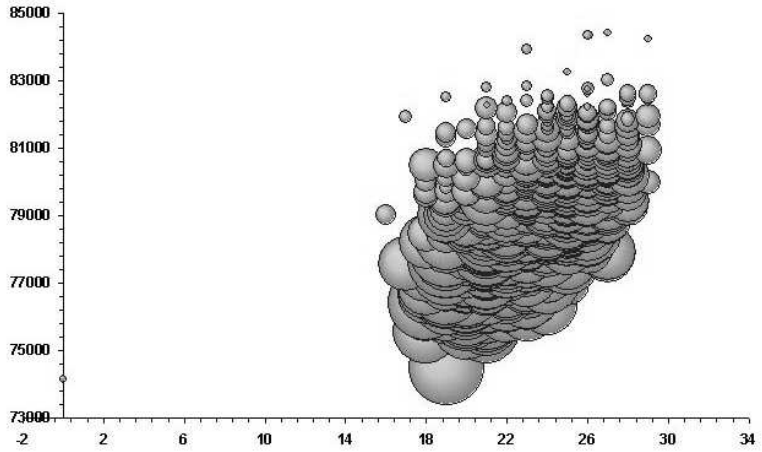

Figure 10. Analysis of local optima for the class $Uniform$

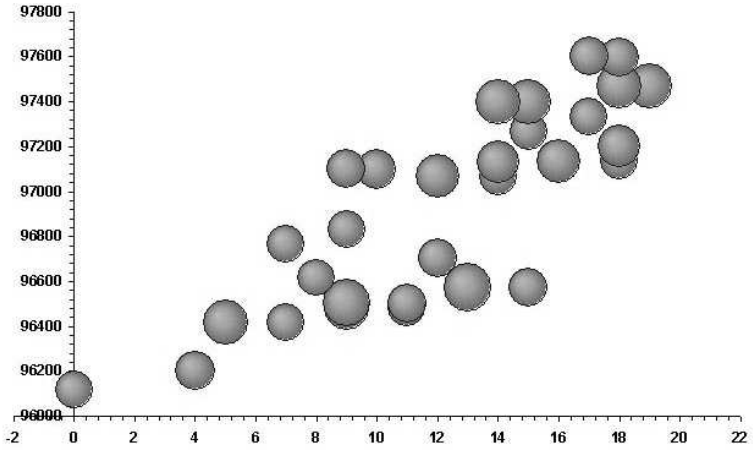

Figure 11. Analysis of local optima for the class *Euclidean*

correlation between the density of local optima in the feasible domain and the objective function. Column R_{100} shows the minimal radius for the 100 biggest spheres. This column indicates that the classes $Gap-A$, $Gap-B$, $Gap-C$ have many large spheres. Class FPP_{11} has spheres that are not so large. In Figures 4, 5, 9, and 10 there are many small spheres quite far from the global optimum. If we imagine the set of local optima as a galaxy then we observe many small spheres at the border of the galaxy and a lot of large spheres at the center. The region of high concentration of local optima consists of the spheres with high and low values of the objective function. The global optimum can be located at the center part of the galaxy and has a large sphere (see Figures 4, 5) or very far from the center and has a small sphere (Figures 6, 10). We cannot predict its radius or place in the galaxy. We can only observe that easy classes have a small galaxy, $N = 1018$ for class $Uniform$ and $N = 40$ for class *Euclidean*. The difficult classes have large galaxies, $N \approx 9000$ for classes BPC_7, FPP_{11}, $Gap-C$. It is possible that the most difficult classes correspond to the cases, where several galaxies are separated from each other by regions with high values of the objective function. Test instances with this property are very interesting for future research [83]. Theoretical properties of different landscapes for NP-hard combinatorial problems can be found in [84].

Table 3 shows the frequency of finding an optimal solution using the following metaheuristics: Probabilistic Tabu Search (PTS), Genetic Algorithm (GA) and Greedy Randomizes Adaptive Search Procedure with Local Improvement ($GRASP + LI$). The stopping criterion for the algorithms is the maximal num-

Table 3. Frequency of obtaining optimal solutions by metaheuristics

Benchmarks classes	Dimension	PTS	GA	GRASP+LI
BPC_7	128	0.93	0.90	0.99
CB_4	144	0.99	0.88	0.68
FPP_{11}	133	0.67	0.46	0.99
$Gap - A$	100	0.85	0.76	0.87
$Gap - B$	100	0.59	0.44	0.49
$Gap - C$	100	0.53	0.32	0.42
$Uniform$	100	1.0	1.0	1.0
$Euclidean$	100	1.0	1.0	1.0

ber of steps by the neighborhood $Flip \cup Swap$. We use number 10^4 as the criteria. The genetic algorithm uses local improvements for each offspring during the evolution. Table 3 indicates that classes $Euclidean$ and $Uniform$ are easy. The average number of steps before an optimal solution is reached is less than 10^3 for all algorithms.

6. Conclusions

We have presented some discrete facility location models, as well as theoretical and experimental results for local search methods. We have explained why the concept of local optimality is important for the theory of computational complexity and numerical methods. There are many open questions in this area. For example, the well-known set covering problem can be reformulated as a facility location problem. We still know very little about the complexity of the corresponding local search problems. In [85] a one-to-one correspondence between Nash equilibria in a facility location game and local optima in a special facility location model is presented. This property helps to determine the computational complexity of finding Nash equilibrium [21]. For the theory of approximation algorithms the concept of local optimality plays an important role as well. In regards to combinatorial optimization problems, the property that all local optima have guaranteed quality with respect to global optima is the basis of the approximability of a large class of problems [75].

Acknowledgments

Many thanks to Vašek Chvătal and Mark Goldsmith for their efforts for improving the presentation of the paper. This work was supported by RFBR grants 09-01-00059, 09-06-00032.

References

[1] B.P. Mirchandani, R.L. Francis (Eds.), *Discrete Location Theory*, Wiley-Interscience, 1990.

[2] Z. Drezner , K. Klamroth, A. Schobel, G. Wesolowsky, The Weber problem, in: Z. Drezner, H. Hamacher (Eds.), *Facility Location. Applications and Theory*, Springer, 2004, 1–36.
[3] M.K. Balinski, P. Wolfe, *On Benders decomposition and a plant location problem*, Working paper ARO-27, Mathematica, Princeton, 1963.
[4] M.A. Efroymson, T.L. Ray, A branch and bound algorithm for plant location, *Operations Research* **14** (1966), 361–368.
[5] A.A. Ageev, V.L. Beresnev, Polynomially solvable cases of simple plant location problem, *Proceedings of the 1st Integer Programming and Combinatorial Optimization Conference IPCO 1990*, Waterloo, 1990, 1–6.
[6] D.S. Hochbaum, Heuristics for the fixed cost median problem *Mathematical Programming* **22** (1982), 148–162.
[7] Y. Mahdian, M. Ye, J. Zhang, Improved approximation algorithms for metric facility location problems, *Proceedings of the 5th International Workshop on Approximation Algorithms for Combinatorial Optimization*, LNCS **2462** (2002), 229–242.
[8] S. Guha, S. Khuller, Greedy strikes back: improved facility location algorithms, *Journal of Algorithms* **31** (1999), 228–248.
[9] B. Korte, J. Vygen, *Combinatorial Optimization. Theory and Algorithms*, Springer, 2005.
[10] B.M. Khumawala, An efficient branch and bound algorithm for the warehouse location problem, *Management Science* **18** (1972), 718–731.
[11] V.L. Beresnev, E.Kh. Gimadi, V. T. Dement'ev, *Extremal Standardization Problems*, Novosibirsk, Nauka, 1978 (in Russian).
[12] S.S. Lebedev, M.I. Kovalevskaya, The Lagrange multiplieres in the simple plant location problem, in: *Researches in Discrete Optimization*, Moscow, Nauka, 1976, 170–180 (in Russian).
[13] V.S. Mikhalevich, V.A. Trubin, N.Z. Shor, *Optimization Problems for Production–Transportation Planing*, Moscow, Nauka, 1986 (in Russian).
[14] O. Bilde, J. Krarup, Sharp lower bounds and efficient algorithms for the simple plant location problem, *Annals of Discrete Mathematics* **1** (1977), 79–97.
[15] D. Erlenkotter, A dual-based procedure for uncapacitated facility location, *Operations Research* **26** (1978) 992–1009.
[16] P. Avella, A. Sassano, I. Vasilev, Computational study of large-scale p-median problems, *Mathematical Programming* **109(1)** (2006), 89–114.
[17] F. Barahona, F. Chudak, Solving large scale uncapacitated facility location problems, in: P. Pardalos (ed.) *Approximation and Complexity in Numerical Optimization*, Kluwer Academic Publishers, Norwell, MA, 2000.
[18] P.L. Hammer, S. Rudeanu, *Boolean Methods in Operations Research and Related Areas*, Springer-Verlag, 1968.
[19] P.L. Hammer, Plant location — a pseudo-Boolean approach, *Israel Journal of Technology* **6** (1968) 330–332.
[20] V.L. Beresnev, Algorithms for minimization of polynomials in Boolean variables, *Problemy Kibernetiki* **36** (1979), 225-246 (in Russian).
[21] Y. Kochetov, A. Kononov, A. Plyasunov, Competitive facility location models, *Computational Mathematics and Mathematical Physics* **49(6)** (2009), 994–1009.
[22] A. Schrijver, *Combinatorial Optimization. Polyhedra and Efficiency*, Berlin, Springer, 2003.
[23] J. Krarup, P.M. Pruzan, The simple plant location problem: survey and synthesis, *European Journal of Operational Research* **12** (1983), 36-81.
[24] E. Boros, P.L. Hammer, Pseudo-Boolean optimization, *Discrete Applied Mathematics* **123** (2002), 155–225.
[25] G.L. Nemhauser, L.A. Wolsey, *Integer and Combinatorial Optimization*, John Wiley & Sons, 1988.
[26] K.I. Aardal, F.A. Chudak, D.B. Shmoys, A 3-approximation algorithm for the k-level uncapacitated facility location problem. *Information Processing Letters* **72** (1999), 161-167.
[27] A. Ageev, Y.Ye, J. Zhang, Improved combinatorial approximation algorithms for the k-level facility location problem, *Proceeding of Automata, Languages and Programming,*

30th International Colloquium, ICALP 2003, LNCS **2719** (2003), 145-156.

[28] E. Goncharov, Branch and bound algorithm for the two-level uncapacitated facility location problem, *Discrete Analysis and Operations Research, ser.2* **5** (1998), 19–39 (in Russian).

[29] D.W. Tcha, B.I. Lee, A branch-and-bound algorithm for the multi-level uncapacitated facility location problem, *European Journal of Operational Research* **18** (1984), 35–43.

[30] E.Kh. Gimadi, Effective algorithms for solving multi-level plant location problems, *Operations Research and Discrete Analysis*, Kluwer Academic Publishers, Dordrecht, 1997, 51–69.

[31] E.Kh. Gimadi, Exact algorithm for some multi-level location problems on a chain and a tree, *Operations Research Proceedings 1996*, Springer-Verlag, Berlin, 1997, 72–77.

[32] P. Hanjoul, D. Peeters, A facility location problem with clients' preference orderings, *Regional Science and Urban Economics* **17** (1987), 451–473.

[33] A.I. Barros, M. Labbe, A general model for the uncapacitated facility and depot location problem, *Location Science* **2** (1994), 173-191.

[34] S. Dempe, *Foundations of Bilevel Programming*, Kluwer Academic Publishers, Dordrecht, 2002.

[35] L.E. Gorbachevskaya, Polynomially solvable and NP-hard bilevel standardization problems, *Ph.D. Thesis, Sobolev Institute of Mathematics*, Novosibirsk, 1998 (in Russian).

[36] P. Hansen, Yu. Kochetov, N. Mladenović, Lower bounds for the uncapacitated facility location problem with user preferences, *Preprint G-2004-24, Mart 2004, GERAD-HEC, Montreal, Canada.* 2004.

[37] J.M.W. Rhys, A selection problem of shared fixed costs and network flows, *Management Science* **17** (1970) 200–207.

[38] E. Alekseeva, Y. Kochetov, Genetic local search for the p-median problem with user preferences. *Discrete Analysis and Operations Research. Series 2*, **14(1)** (2007), 3-31 (in Russian).

[39] L. Canovas, S. Garcia, M. Labbe, A. Marin, A strengthened formulation for the simple plant location problem with order, *Operations Research Letters* **35(2)** (2007), 141–150.

[40] I. Vasil'ev, K. Klimentova, Y. Kochetov, New lower bounds for the facility location problem with user preferences *Computational Mathematics and Mathematical Physics* **49(6)** (2009), 1055–1066.

[41] F. Plastria, L. Vanhaverbeke, Discrete models for competitive location with foresight, *Computers and Operations Research*, **35** (2008), 683–700.

[42] C.M.C. Rodriguez , J.A.M. Perez, Multiple voting location problems, *European Journal of Operational Research* **191** (2008), 437-453.

[43] S.L. Hakimi, Locations with spatial interactions: competitive locations and games, in: P. Mirchandani, R. Francis (Eds.) *Discrete Location Theory*, Wiley, 1990, 439–478.

[44] E.V. Alekseeva, N.A. Kochetova, Y.A. Kochetov, A.V. Plyasunov, *A heuristic and exact methods for the discrete* $(r|p)$*-centroid problem.* LNCS **6022** (2010), 11-22.

[45] H. Noltemeier, J. Spoerhase, H.C. Wirth, Multiple voting location and single voting location on trees. *European Journal of Operational Research* 181 (2007), 654–667.

[46] J. Spoerhase, H.C. Wirth, (r,p)-Centroid problems on paths and trees, *Tech. Report 441, Inst. Comp. Science, University of Wurzburg*, 2008.

[47] J. Bhadury, Y. Eiselt, J. Jaramillo, An alternating heuristic for medianoid and centroid problems in the plane, *Computers and Operations Research* **30** (2003), 553–565.

[48] E. Alekseeva, N. Kochetova, Y. Kochetov, A. Plyasunov, A hybrid memetic algorithm for the competitive p-median problem, *Proceedings of INCOM 2009, Moscow, June 3-5, 2009.*

[49] E. Alekseeva, N. Kochetova, Upper and lower bounds for the competitive p-median problem, *Proceedings of XIV Baikal International School-Seminar*, **1** (2008), 563-569 (in Russian).

[50] S. Benati, G. Laporte, Tabu search algorithms for the $(r|X_p)$–medianoid and $(r|p)$–centroid problems, *Location Science* **2(4)** (1994), 193–204.

[51] A.Plyasunov, Personal communication.

[52] E. Alekseeva, Y. Kochetov, A. Plyasunov, Complexity of local search for the p-median

problem, *European Journal of Operational Research* **191** (2008), 736–752.

[53] L. K. Grover, Local search and the local structure of NP-complete problems, *Operations Research Letters*, **12 (4)** (1992), 235–244.

[54] G. Gutin, Exponential neighborhood local search for the traveling salesman problem, *Computers & Operations Research* **26** (1999), 313–320.

[55] G. Gutin, A. Yeo, Small diameter neighborhood graphs for the traveling salesman problem: four moves from tour to tour, *Computers & Operations Research*, **26** (1999), 321–327.

[56] R.K. Ahuja, O.E. James, B. Orlin, and A.P. Punnen, A survey of very large-scale neighborhood search techniques, *Discrete Applied Mathematics*, **123** (2002), 75-102.

[57] A.A. Schäffer, M. Yannakakis, Simple local search problems that are hard to solve, *SIAM Journal on Computing* **20** (1991), 56–87.

[58] D.C. Johnson, C.H. Papadimitriou, M. Yannakakis, How easy is local search? *Journal of Computer and System Sciences* **37** (1988), 56–87.

[59] M.W. Krentel, Structure in locally optimal solutions, *30th Annual Symposium on Foundations of Computer Science*, IEEE Comput. Soc. Press, Los Alamitos, CA, 1989, 216–222.

[60] C.H. Papadimitriou, The complexity of the Lin–Kernighan heuristic for the traveling salesman problem, *SIAM Journal on Computing* **21** (1992), 450–465.

[61] M.W. Krentel, On finding and verifying locally optimal solutions, *SIAM Journal on Computing* **19** (1990), 742–751.

[62] J.J. Hopfield, Neural networks and physical systems with emergent collective computational abilities, *Proceedings of the National Academy of Sciences of the USA* **79** (1982), 2554–2558.

[63] Yu. Kochetov, D. Ivanenko, Computationally difficult instances for the uncapacitated facility location problem, in: Ibaraki T. et al. (Eds.) *Metaheuristics: Progress as Real Solvers*, Springer, 2005, 351–367.

[64] M. Yannakakis, Computational complexity, in: E. Aarts, J.K. Lenstra (Eds.) *Local Search in Combinatorial Optimization*, Chichester: John Wiley & Sons, 1997, 19–55.

[65] Yu. Kochetov, M. Paschenko, A. Plyasunov, On complexity of local search for the p-median problem, *Discrete Analysis and Operations Research, Series 2*, **12** (2005) 44–71 (in Russian).

[66] T. Vredeveld, J.K. Lenstra, On local search for the generalized graph coloring problem, *Operations Research Letters* **31** (2003), 28–34.

[67] J. Dreo, P.Siarry, A. Petrowski, E. Taillard, *Metaheuristics for Hard Optimization: Methods and Case Studies*, Springer, 2006.

[68] F. Glover, M. Laguna, *Tabu Search*, Kluwer Academic Publishers, Dordrecht, 1997.

[69] P. Hansen, N. Mladenović, Variable neighborhood search: principles and applications (invited review), *European Journal of Operational Research* **130** (2001), 449–467.

[70] C. Tovey, On the number of iterations of local improvement algorithms, *Operations Research Letters*, **2** (1983), 231–238.

[71] C. Tovey, Hill climbing with multiple local optima, *SIAM Journal on Algebraic and Discrete Methods*, **6** (1985), 384–393.

[72] C. Tovey, Local improvement on discrete structures, in: E. Aarts, J.K. Lenstra (Eds.) *Local Search in Combinatorial Optimization*, Chichester: John Wiley & Sons, 1997, 57–89.

[73] J.B. Orlin, A.P. Punnen, A. Schulz, Approximate local search in combinatorial optimization, *SIAM Journal of Computing* **33** (2004), 1201–1214.

[74] C.H. Papadimitriou, K. Steiglitz, On the complexity of local search for the traveling salesman problem, *SIAM Journal of Computing* **6** (1977), 76–83.

[75] G. Ausiello, M. Protasi, Local search, reducibility and approximability of NP optimization problems, *Information Processing Letters* **54** (1995), 73–79.

[76] M.G.C. Resende, R.F. Werneck, A hybrid multistart heuristic for the uncapacitated facility location problem, *European Journal of Operational Research* **174** (2006), 54–68.

[77] Yu. Kochetov, E. Alekseeva, T. Levanova, M. Loresh, Large neighborhood local search for the p-median problem, *Yugoslav Journal of Operations Research* **15** (2005), 53–63.

[78] T.V. Levanova, M.A. Loresh, Algorithms of ant system and simulated annealing for the p-median problem, *Automation and Remote Control* **65** (2004), 431–438.

[79] N. Mladenović, J. Brimberg, P. Hansen, J. Moreno-Perez, The p-median problem: A survey

of metaheuristic approaches, *European Journal Operational Research* **179** (2007), 927–939.
[80] K.D. Boese, A.B. Kahng, S. Muddu, A new adaptive multi-start technique for combinatorial global optimization, *Operations Research Letters* **16** (1994), 101–113.
[81] M. Jr. Hall, *Combinatorial Theory*, Blaisdell, Waltham, MA, 1967.
[82] D. Krotov, Lower bounds for number of m-quasi groups of order 4 and number of perfect binary codes, *Discrete Analysis and Operations Research* **7** (2000), 47–53 (in Russian).
[83] M. Qasem, A. Prügel-Bennett, Learning the large–scale structure of the MAX–SAT landscape using populations, *IEEE Transactions on Evolutionary Computation*, **14 (4)** (2010), 518–529.
[84] E. Angel, V. Zissimopoulos, On the classification of NP-complete problems in terms of their correlation coefficient, *Discrete Applied Mathematics*, **99** (2000), 261–277.
[85] E. Tardos, T. Wexler, Network formation games and the potential function method, in: N. Nisan, T. Roughgarden, E. Tardos, V. V. Vazirani (Eds.) *Algorithmic Game Theory*, Cambridge University Press, 2007, 487–516.