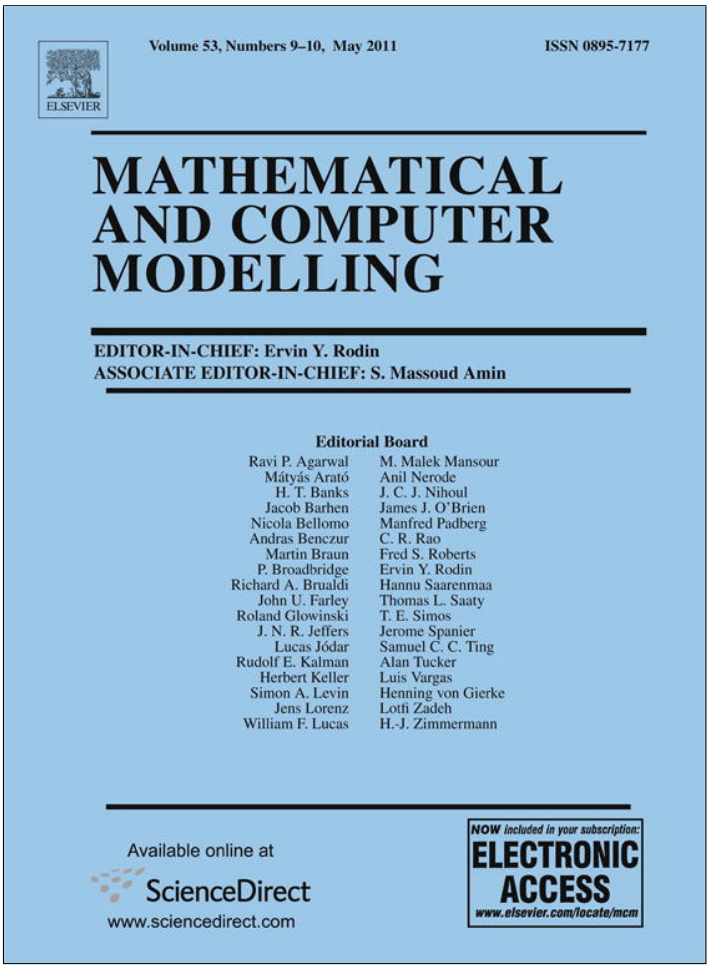

Mathematical and Computer Modelling 53 (2011) 1719–1736

Contents lists available at ScienceDirect

Mathematical and Computer Modelling

journal homepage: www.elsevier.com/locate/mcm

Complexity evaluation of benchmark instances for the *p*-median problem

B. Goldengorin [a,b,*], D. Krushinsky [b]

[a] *Department of Applied Mathematics and Informatics, Higher School of Economics, Nizhny Novgorod Branch, B. Pecherskaya 25/12, Nizhny Novgorod, 603155, Russian Federation*

[b] *University of Groningen, Department of Operations, P.O. Box 800, 9700 AV Groningen, The Netherlands*

ARTICLE INFO

Article history:
Received 6 December 2009
Received in revised form 22 December 2010
Accepted 22 December 2010

Keywords:
p-Median problem
Pseudo-Boolean polynomial
Problem size reduction
Data complexity

ABSTRACT

The paper is aimed at experimental evaluation of the complexity of the *p*-Median problem instances, defined by $m \times n$ costs matrices, from several of the most widely used libraries. The complexity is considered in terms of possible problem size reduction and preprocessing, irrespective of the solution algorithm. We use a pseudo-Boolean representation of PMP instances that allows several reduction techniques to be applied in a straightforward way: combining similar monomials in the polynomial, truncation of the polynomial from degree $(m-1)$ to $(m-p)$ implying costs matrix truncation and exclusion of some rows from the costs matrix (preprocessing based only on compactification of the costs matrix), decomposition of the polynomial into the minimum number of expressions inducing the minimum number of aggregated columns (reduction of the columns' number in the costs matrix). We show that the reduced instance has at most $\sum_{i=1}^{m-p} \min\{n, \binom{m}{i}\}+1$ nonzero entries. We also provide results of computational experiments with the mentioned reductions that allow classification of the benchmark data complexity. Finally, we propose a new benchmark library of instances not amenable to size reduction by means of data compactification.

1. Introduction

The *p*-Median Problem (PMP) is a well-known problem within minisum location–allocation problems. A detailed introduction to this problem and solution methods appear in [1,2]. For a directed weighted graph $G = (V, A, C)$, with number of vertices $|V|$, set of arcs $(i, j) \in A \subseteq V \times V$, and weights (distances, similarities, etc.) $C = \{c(i, j) : (i, j) \in A\}$, the PMP consists of determining p nodes (the *median nodes*, $1 \leq p \leq |V|$) minimizing the total sum of weights to all other nodes of the graph.

Further, for the sake of clarity, we will follow the terminology inherited from location–allocation applications and represent the set of vertices V as a union of two (possibly intersecting) sets I and J, such that $|I| = m$, $|J| = n$. We will call the elements of I locations and those of J — clients. Moreover, we treat weights $c(i, j)$ as the costs of serving client j $(j \in J)$ from location i $(i \in I)$. It should be mentioned that for the benchmark instances that we used initially $m = n$, but two of our reduction techniques break this balance.

The PMP is NP-hard [3], and has many applications in location (see [4] and references within) and clustering analysis (see e.g., [5] and references within). A recent computational study by Avella et al. [6] shows that PMP instances with $|I \times J| > 360{,}000$ cannot be uploaded by commercial Mixed Integer Programming (MIP) codes, mainly due to memory restrictions. For example, CPLEX could not load the instance pmed40 from Beasley's OR-Library ($|V| = 900$) [7], due to

* Corresponding author at: University of Groningen, Department of Operations, P.O. Box 800, 9700 AV Groningen, The Netherlands.
E-mail addresses: b.goldengorin@rug.nl (B. Goldengorin), d.krushinsky@rug.nl (D. Krushinsky).

0895-7177/$ – see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.mcm.2010.12.047

its size (see [6]). These PMP instances (OR instances) are used widely (see e.g., [1,2,4,6,8] and references within) with the purpose of comparing the computational efficiency of exact and heuristic algorithms for solving PMP. This paper is focused on estimating the complexity of PMP instances in terms of possible size reduction. Our definition of the complexity measure is universal in the sense that it is not biased to any specific solution algorithm and we claim that if some instance has high complexity in our terms then it will be complex for any solution algorithm (existing or forthcoming).

Problem size reduction is a very common technique in integer programming and combinatorial optimization. It can be used to find a compact representation of PMP instances, see, for example, [9–15]. Classical reductions of PMP instances are based either on reduction tests (see e.g., [16]) or on good lower bounds (see e.g., [8]). In this paper, we present three classes of reductions of a PMP instance using its pseudo-Boolean formulation due to Hammer [17] and Beresnev [18]. The first reductions are aimed at minimization of the number of terms in the pseudo-Boolean polynomial by reducing similar monomials and truncation from degree $(m - 1)$ to $(m - p)$. Reduction of similar monomials is indicated in [18–22]. An analogue of truncation is used in [16]. The second technique reduces the number of columns in the costs matrix. It is based on covering the Hasse diagram, comprising terms of the polynomial, with the minimum number of chains. The third reduction is a preprocessing that allows exclusion of some rows of the costs matrix from consideration as they do not contribute to the value of optimal solutions.

All our reductions are aimed at a compact representation of instance data. Even though they sometimes provide partial solutions as a side effect, this should be considered as a consequence of redundancy in the instance data, rather than a result of our attempts to find an optimal solution which is beyond the scope of this paper.

For the numerical experiments with mentioned reductions we used benchmark instances from the four most popular libraries: OR, TSP, ODM, RW. The first one, the OR library, was introduced by Beasley [23] and is available at [7]. Every node is both a potential location and a client, and the costs are the lengths of the shortest paths between the corresponding nodes.

The TSP library was originally proposed for the traveling salesman problem (TSP) and is available at [24]. TSP instances are defined as sets of points in a two dimensional plane. Every point is considered both a potential location and a client, and the costs are simply Euclidean distances.

Instances from the next library that we studied are based on the optimal diversity management (ODM) problem. For the description of this problem and instances see [8].

Finally, we considered instances proposed by Resende and Werneck [25]. These problems are defined on random distance matrices. In every case the number of potential facilities m is equal to the number of clients n and distances are integers taken uniformly at random from the interval $[1, n]$. The library contains five instances with $n = 100, 200, 250, 500, 1000$.

The rest of the paper is organized as follows. In Section 2 we define the notion of instance data complexity and provide a brief overview of the existing approaches to reducing the problem size. In Section 3 we formulate the pseudo-Boolean representation of the p-Median problem. In Section 4 our reduction techniques based on reducing similar monomials in the pBp [18–22], its truncation and decomposition into the minimal number of columns are presented. In that section we also prove that pseudo-Boolean formulation allows the most compact representation of the instance. Next, in Section 5 we describe an approach to preprocessing PMP instances by excluding some rows of the costs matrix that do not contribute to the value of optimal solutions. Two latter sections contain results of computational experiments on benchmark instances. In Section 6 we formulate the criteria of complexity and introduce our benchmark library, the instances of which cannot be reduced by any of the well-known approaches. Section 7 summarizes the obtained results and suggests directions for future research.

2. Data complexity and problem size reduction

Problem size reduction is a very common technique in integer programming and combinatorial optimization that can be used to find a compact representation of PMP instances. It is aimed at constructing an instance of smaller size that is assumed to be more easily solvable and provide an optimal solution to the initial instance. Moreover, it is straightforward that if the procedure of size reduction is as time-consuming as the procedure for solving the initial problem, it has no sense. These considerations lead to the following definition.

Definition 1. We will call instance $\mathcal{D}$ a *reduced version* of instance $\mathcal{C}$ ($\mathcal{D} = red(\mathcal{C})$) if it satisfies the following conditions:

1. $\emptyset \subset opt.solutions(\mathcal{D}) \subseteq opt.solutions(\mathcal{C})$
2. $size(\mathcal{D}) \leq size(\mathcal{C})$
3. $\mathcal{D}$ can be obtained from $\mathcal{C}$ in polynomial time.

The first requirement guarantees that by solving $\mathcal{D}$ to optimality one immediately obtains an optimal solution to $\mathcal{C}$ (here we assume the feasibility of $\mathcal{C}$), while the second one is related to the reduction itself. Finally, the last requirement is needed to make the definition useful in practice: if for some NP-hard problem computing the reduced instance $\mathcal{D}$ is as hard as solving $\mathcal{C}$ then such a reduction is senseless. Based on this definition of a reduced instance we define complexity of the instance data in the following way.

Definition 2. By *complexity* of the instance data $\mathcal{C}$ (relative to a particular problem) we mean the minimum capacity of the storage needed to be able to obtain an optimal solution to the initial instance:

$comp(\mathcal{C}) = \min\{size(\mathcal{D}) : \ \mathcal{D} = red(\mathcal{C})\}$.

It should be noticed that without a reference to a particular problem (in our case — the p-Median problem) this definition is meaningless. However, even when the problem is fixed, it provides neither a direct way for constructing a compact representation of the data, nor even for determining the minimum required space. Further we briefly describe existing approaches to reducing the problem size and thus to obtaining upper bounds of instance data complexity.

As the costs matrix of a PMP instance has $m \times n$ elements, it is clear that this value is the most trivial upper bound for $comp(\mathcal{C})$. This value is achieved by the classical ILP representation (see [26]) of the p-Median problem with its objective function defined as:

$$\sum_{j\in J}\sum_{i\in I} c_{ij}x_{ij}. \tag{1}$$

Here c_{ij} denote entries of the costs matrix and x_{ij} are decision variables ($x_{ij} = 1$ if the jth client is served from the ith location, otherwise $x_{ij} = 0$). Cornuejols et al. [19] introduced an alternative formulation of the problem. For any client j, let K_j be the number of different distances from j to any location. It follows that $K_j \leq m$. Let $D_j^1 < D_j^2 < \cdots < D_j^{K_j}$ be these distances, sorted. For each client j it is possible to define a hierarchy of neighbourhoods V_j^k such that each V_j^k is a set of locations within the distance D_j^k from client j. Naturally, in an optimal solution a client j is assigned to its neighbourhood with the smallest D_j^k containing the opened location. Thus, instead of x_{ij} this formulation uses variables z_j^k such that $z_j^k = 1$ if and only if there are no opened locations in V_j^k. The objective function in this case is defined as:

$$\sum_{j\in J}\left(D_j^1 + \sum_{k=1}^{K_i-1}(D_i^{k+1} - D_i^k)z_i^k\right). \tag{2}$$

Informally, this representation implies that only different elements in each column of the costs matrix are meaningful and the problem size can be reduced by storing only the pairwise different elements from each column. A further reduction is proposed in [20]. It states that if for some j, k, j', k' holds $V_j^k = V_{j'}^{k'}$, then for any feasible solution $z_j^k = z_{j'}^{k'}$ and some terms in (2) can be merged. Several reductions are also presented in [21], but they are similar to those described above. There are also some papers aimed at reduction of the number of constraints in the ILP formulation of the problem (see e.g. [16,27]); however, the number of coefficients in the objective function remains the same.

It should be noticed that most of the reduction techniques described in the literature are based on ILP formulation of the p-Median problem and apply artificial tricks exploiting some features of the instance. On the contrary, in this study we use a different – pseudo-Boolean – formulation of the problem and show that this representation itself naturally leads to several reductions that allow us to obtain better estimates of the instance data complexity and include all known reductions.

3. Pseudo-Boolean representation

Recall that given sets $I = \{1, 2, \ldots, m\}$ of sites in which plants can be located, $J = \{1, 2, \ldots, n\}$ of clients, a matrix $C = [c_{ij}]$ of transportation costs (supplying costs, distances, similarities, etc.) for each $j \in J$ from each $i \in I$, the number p of plants to be opened, and a unit demand at each client site, the p-Median Problem (PMP) is one of finding a set $S \subseteq I$ with $1 \leq |S| = p \leq m$, such that the total costs

$$f_C(S) = \sum_{j\in J}\min\{c_{ij}|i \in S\} \tag{3}$$

of satisfying all unit demands is minimized. Note that non-unit demands $d_j \neq 1$ can be scaled by $c'_{i,j} = c_{i,j}d_j$, and the number of served clients by each plant is unbounded (the so called *uncapacitated* location problem, see e.g., [1,4]). An instance of the problem is described by an $m \times n$ matrix $C = [c_{ij}]$ and the number $1 \leq p \leq |I|$. We assume that entries of C are nonnegative and finite, i.e. $C \in \mathbb{R}_+^{mn}$. The PMP is thus the problem of finding

$$S^\star \in \arg\min\{f_C(S) : \emptyset \subset S \subseteq I,\ |S| = p\}. \tag{4}$$

In this section we are going to reformulate the objective function $f_C(S)$ of the PMP (4) in terms of a pseudo-Boolean polynomial (due to Beresnev [18]). It is enough to find a pseudo-Boolean representation for each addend $\min\{c_{ij}|i \in S\}$, and sum up addends for all $j \in J$. The smallest value of $\min\{c_{ij}|i \in S\}$ is attained if $S = I$, i.e. the smallest value is chosen among all entries c_{ij} for a fixed column j. It is clear that the unit demand of column j cannot be satisfied more cheaply than this smallest value. Assume that this smallest value is attained at an entry $c_{\pi_{1j}j}$ of column j such that π_{1j} indicates the number of the row containing this smallest entry $c_{\pi_{1j}j}$ in a column j. In terms of the original PMP, if the site numbered by π_{1j} is open, then the unit demand of client j will be satisfied by costs $c_{\pi_{1j}j}$, otherwise (if the site π_{1j} is closed, but all other sites in $I \setminus \{\pi_{1j}\}$ are opened) the cheapest way to satisfy the unit demand of client j is by the value of a second smallest entry $c_{\pi_{2j}j}$. The value of a second smallest entry $c_{\pi_{2j}j}$ can be represented as follows: $c_{\pi_{2j}j} = c_{\pi_{1j}j} + [c_{\pi_{2j}j} - c_{\pi_{1j}j}]$. Similarly, if both sites π_{1j}, π_{2j} are closed and all other sites are opened, then the unit demand of client j will be satisfied by the value of a third smallest entry $c_{\pi_{3j}j} = c_{\pi_{1j}j} + [c_{\pi_{2j}j} - c_{\pi_{1j}j}] + [c_{\pi_{3j}j} - c_{\pi_{2j}j}]$, etc. In other words, depending on the set of opened and closed sites from I the

corresponding smallest value of $\min\{c_{i,j}|i \in S\}$ can be represented by the sum of the smallest values of entries in column j and the corresponding differences of ordered entries in column j. By introducing a Boolean variable $y_{\pi_{1j}} = 0$ if the site π_{1j} is opened, and $y_{\pi_{1j}} = 1$ if the site π_{1j} is closed, we are able to express, for example, the costs of satisfying the unit demand j depending on whether the site π_{1j} is opened or closed (if π_{1j} is closed then we assume that π_{2j} is open, i.e., $y_{\pi_{2j}} = 0$), as follows: $c_{\pi_{2j}j} = c_{\pi_{1j}j} + [c_{\pi_{2j}j} - c_{\pi_{1j}j}]y_{\pi_{1j}}$.

To illustrate, let us consider the following first column C^1 of matrix C, namely $C^1 = (c_{11}, c_{21}, c_{31}, c_{41})^{\mathrm{T}} = (7, 10, 16, 11)^{\mathrm{T}}$. After ordering its entries in a non-decreasing order $7 < 10 < 11 < 16$ we have that the corresponding permutation is $\Pi^1 = (1, 2, 4, 3)^{\mathrm{T}}$. If the Boolean vector (y_1, y_2, y_3, y_4) reflects an opened (closed) plant at site $i = 1, 2, 3, 4$, then depending on the set of opened plants $S \subseteq \{1, 2, 3, 4\}$, the smallest value of $\min\{c_{i1}|i \in S\} = [7 + 3y_1 + 1y_1y_2 + 5y_1y_2y_4]$. For example, if $S = \{2, 4\}$, then $\mathbf{y} = (1, 0, 1, 0)$, and

$$\min\{c_{i1}|i \in \{2, 4\}\} = 7 + 3 \times 1 + 1 \times 1 \times 0 + 5 \times 1 \times 0 \times 0 = 10. \tag{5}$$

Now we are ready to introduce a pseudo-Boolean polynomial (pBp) related to each addend $\min\{c_{i,j}|i \in S\}$ by the following definitions.

An $m \times n$ *ordering matrix* $\Pi = [\pi_{ij}]$ is a matrix of which each column $\Pi^j = (\pi_{1j}, \ldots, \pi_{mj})^{\mathrm{T}}$ defines a permutation of $1, \ldots, m$, reflecting an ordering of row numbers, for each column in C induced by a non-decreasing ordering of its entries such that $c_{\pi_{1j}j} \leq c_{\pi_{2j}j} \leq \cdots \leq c_{\pi_{mj}j}$, for $j = 1, \ldots, n$,. There may exist several orderings of a column j if it has equal entries, and hence several ordering matrices for a given instance of the PMP. Given a matrix C, the set of all ordering matrices Π such that $c_{\pi_{1j}j} \leq c_{\pi_{2j}j} \leq \cdots \leq c_{\pi_{mj}j}$, for $j = 1, \ldots, n$, is denoted by $perm(C)$.

Corresponding to an ordering matrix $\Pi = [\pi_{ij}]$, we define differences between the costs for each $j \in J$ as

$$\Delta c[0, j] = c_{\pi_{1j}j}, \quad \text{and} \tag{6}$$

$$\Delta c[k, j] = c_{\pi_{(k+1)j}j} - c_{\pi_{kj}j}, \quad k = 1, \ldots, m - 1. \tag{7}$$

These differences can be arranged into an $m \times n$ *differences matrix* that we denote by Δ.

Defining

$$y_i = y_i(S) = \begin{cases} 0 & \text{if } i \in S, \\ 1 & \text{otherwise,} \end{cases} \text{ for each } i = 1, \ldots, m \tag{8}$$

$$\begin{aligned} \min\{c_{ij}|i \in S\} &= \Delta c[0, j] + \Delta c[1, j] \cdot y_{\pi_{1j}} + \Delta c[2, j] \cdot y_{\pi_{1j}} \cdot y_{\pi_{2j}} + \cdots + \Delta c[m - 1, j] \cdot y_{\pi_{1j}} \cdots y_{\pi_{(m-1)j}} \\ &= \Delta c[0, j] + \sum_{k=1}^{m-1} \Delta c[k, j] \cdot \prod_{r=1}^{k} y_{\pi_{rj}}, \end{aligned}$$

we can indicate any solution S by a vector $\mathbf{y} = (y_1, y_2, \ldots, y_m)$. Its total costs given by $\mathbf{y}$ corresponding to an ordering matrix $\Pi \in perm(C)$ is

$$\mathcal{B}_{C,\Pi}(\mathbf{y}) = \sum_{j=1}^{n} \left\{ \Delta c[0, j] + \sum_{k=1}^{m-1} \Delta c[k, j] \cdot \prod_{r=1}^{k} y_{\pi_{rj}} \right\}. \tag{9}$$

In [28] it is shown that the pseudo-Boolean polynomial is unique and does not depend on a particular choice of matrix $\Pi \in perm(C)$. This allows us to use a notation $\mathcal{B}_C(\mathbf{y})$ instead of $\mathcal{B}_{C,\Pi}(\mathbf{y})$ without any confusion.

Note that a solution $\mathbf{y}$ is called feasible if $\sum_{i=1}^{m} y_i = m - p$ and a pseudo-Boolean formulation of the PMP is as follows:

$$\min \left\{ \mathcal{B}_C(\mathbf{y}) : \ \mathbf{y} \in \{0, 1\}^m, \ \sum_{i=1}^{m} y_i = m - p \right\}. \tag{10}$$

4. Reduction techniques

4.1. Reduction of the number of monomials in the pBp

Given a variable vector $\mathbf{y} = (y_1, y_2, \ldots, y_m)$, the expressions $\mathcal{T} = \prod_{i \in T} y_i$ and $\alpha\mathcal{T} = \alpha \prod_{i \in T} y_i$ $(T \subseteq \{1, \ldots, m\}, \alpha \in \mathbb{R})$ are called a *term* and a *monomial*, respectively. We also call two monomials *similar* if their terms are identical. Finally, by *reduction of monomials* we mean algebraic summation of similar monomials.

Reduction of the number of monomials in pBp consists of three stages. First, as some locations may have an equal distance to several clients, the corresponding entries in the differences matrix are zero and the number of terms in the polynomial is usually less than mn (see column $\#T$ in Table 1). This reduction is similar to the one introduced by many authors [18–22]

Table 1
Reduction of the pBp for benchmark instances.

Library	Instance	m	Entries in matrix C	$\#T$	$\#T_r$	Reduction (%)
OR	pmed1	100	10,000	7506	6722	32.78
OR	pmed15	300	90,000	20,182	17,428	80.64
OR	pmed26	600	360,000	29,963	25,694	92.86
OR	pmed40	900	810,000	36,326	31,642	96.09
ODM	BN48	42	411	411	329	19.95
ODM	BN1284	1284	88,542	88,447	85,416	3.53
ODM	BN3773	3773	349,524	348,063	341,775	2.22
ODM	BN5535	5535	666,639	665,577	654,709	1.79
TSP	rd100	100	9900	9394	9243	6.63
TSP	D657	657	430,992	368,233	367,355	14.77
TSP	fl1400	1400	1958,600	838,110	836,557	57.29
TSP	pcb3038	3038	9226,406	5763,280	5759,404	37.58
RW	rw100	100	10,000	6357	6232	37.68
RW	rw200	200	40,000	25,351	25,099	37.25
RW	rw250	250	62,500	39,542	39,228	37.24
RW	rw500	500	250,000	158,007	157,362	37.06
RW	rw1000	1000	1000,000	631,805	630,543	36.95

and can be illustrated by the following small example: let $m = 4$, $n = 5$ and the costs matrix is

$$C = \begin{bmatrix} 7 & 15 & 10 & 7 & 10 \\ 10 & 17 & 4 & 11 & 22 \\ 16 & 7 & 6 & 18 & 24 \\ 11 & 7 & 6 & 12 & 8 \end{bmatrix}. \tag{11}$$

A possible permutation matrix and the corresponding difference matrix are

$$\Pi = \begin{bmatrix} 1 & 3 & 2 & 1 & 4 \\ 2 & 4 & 3 & 2 & 1 \\ 4 & 1 & 4 & 4 & 3 \\ 3 & 2 & 1 & 3 & 2 \end{bmatrix} \tag{12}$$

and

$$\Delta = \begin{bmatrix} 7 & 7 & 2 & 7 & 8 \\ 3 & 0 & 2 & 4 & 2 \\ 1 & 8 & 0 & 1 & 4 \\ 5 & 2 & 4 & 6 & 8 \end{bmatrix}. \tag{13}$$

Thus, the pBp is $\mathcal{B}_C = [7 + 3y_1 + 1y_1y_2 + 5y_1y_2y_4] + [7 + 0y_3 + 8y_3y_4 + 2y_1y_3y_4] + [4 + 2y_2 + 0y_2y_3 + 4y_2y_3y_4] + [7 + 4y_1 + 1y_1y_2 + 6y_1y_2y_4] + [8 + 2y_4 + 4y_1y_4 + 8y_1y_3y_4]$. As there are two zeroes in the differences matrix, the initial (in contrast to reduced and truncated) pBp has $mn - 2 = 18$ nonzero terms (we will denote this characteristic by $\#T$).

Second, the pBp can be subjected to reducing similar monomials (by its essence, it corresponds to the second reduction rule from [20], p. 11). In the considered example this procedure leads to a polynomial $33 + 7y_1 + 2y_2 + 2y_4 + 2y_1y_2 + 8y_3y_4 + 4y_1y_4 + 11y_1y_2y_4 + 10y_1y_3y_4 + 4y_2y_3y_4$ with 10 monomials. We denote the number of monomials in such a pBp with reduced similar monomials by $\#T_r$.

Finally, as shown in [28], for any feasible solution $\mathbf{y}$ the value of truncated polynomial $\mathcal{B}_{C,p}$ obtained from $\mathcal{B}_C$ by deleting all terms of degree higher that $(m - p)$ is equal to the value of the initial pBp. For example, $\mathcal{B}_C = 33 + 7y_1 + 2y_2 + 2y_4 + 2y_1y_2 + 8y_3y_4 + 4y_1y_4 + 11y_1y_2y_4 + 10y_1y_3y_4 + 4y_2y_3y_4$ with $p = 2$, i.e. $\mathcal{B}_{C,2} = 33 + 7y_1 + 2y_2 + 2y_4 + 2y_1y_2 + 8y_3y_4 + 4y_1y_4$ has just seven monomials. This makes it possible for the particular problem with a fixed number of medians to truncate the polynomial thus reducing its size to at most $(m - p) \cdot n$.

In order to determine the effect of the techniques mentioned above, a number of experiments with instances from the four libraries were carried out. The results of pseudo-Boolean formulation and the reduction of similar monomials for typical representatives of each library are given in Table 1. We computed the reduction (see the rightmost column of the table) as $(mn - \#T_r)/mn \times 100\%$. As can be seen from the table, instances from the OR library allow the highest reduction of the number of terms in the pBp. For example, for the instance pmed40 the size of the polynomial is about 4% of the number of entries in the costs matrix. So, from the point of view of our notion of complexity, these instances are the easiest ones. Instances from TSP and RW libraries also allow compact representation of the polynomial, while ODM instances are the most complex ones and allow only minor reduction of the number of terms.

Of certain interest is a relation between the instance size and the achieved reduction (rightmost column in Table 1). For OR and TSP libraries this factor tends to increase for larger problems implying that pseudo-Boolean representation is efficient for large instances from these classes. However, for ODM library the situation is opposite, so from this point of

view ODM instances are also hard. With randomized graphs from RW library the reduction ratio is almost constant, so these instances are somewhere in between the previous two groups.

Despite the differences in performance between the above mentioned libraries, truncation of the polynomial has a similar impact on the required space for all the considered instances (resulting in at most $(m-p)\cdot n$ entries). We have observed that nonzero entries are uniformly distributed over the rows of the differences matrix Δ (in other words, the numbers of nonzero monomials of different degrees are approximately the same). This means that with increasing p the number of monomials in the truncated polynomial $\mathcal{B}_{C,p}$ decreases in a linear fashion from $\#T_r$ to 1 (if $p = m$ the polynomial is just a constant). Moreover, if we denote by $p^* \leq m$ the (minimum) number of rows that contain all minima in columns, then the polynomial reduces to a constant for $p \geq p^*$.

It should be mentioned that a pBp can be constructed in polynomial time. Taking into account that all our reductions also have polynomial complexity, the requirements of the definition of reduced instances are satisfied (see Definition 1).

4.2. Reduction of the number of clients (columns)

In order to show why the reduction of the number of clients (columns) is possible we have to give the following definitions.

Definition 3. Two PMP instances defined on costs matrices C and D are called *equivalent* if C and D are of the same size and $\mathcal{B}_{C,p}(\mathbf{y}) = \mathcal{B}_{D,p}(\mathbf{y})$.

Definition 4. Having an $m \times n$ costs matrix C, by *aggregation of clients* (columns) we mean construction of such an $m \times n'$ matrix D that $\mathcal{B}_{C,p}(\mathbf{y}) = \mathcal{B}_{D,p}(\mathbf{y})$ and $n' < n$.

This means that if there exists some costs matrix D that leads to the same polynomial as C and D has fewer columns, then the p-Median problem defined on C can be substituted by the problem defined on D. So, the question is: given a costs matrix C and the number of medians p, find such a matrix D that corresponds to the same truncated polynomial as C and has the minimum possible number of columns.

The idea behind this type of processing is as follows. Each chain of embedded terms in a pBp corresponds to some permutation and a column of differences. At the same time, over the terms of the polynomial it is possible to define a relation of partial order, that, in turn, can be represented by the Hasse diagram. It is clear that all the terms can be covered by n chains that correspond to n columns of the differences matrix. This means that all vertices of the Hasse diagram can be covered by n (internally) vertex disjoint chains. However, the observation that for some instances the reduction of similar monomials leads to a substantial decrease in their number suggests a possibility that all terms can be covered by fewer chains. Having a chain of embedded terms it is possible to reconstruct a permutation and a row of the differences matrix. Thus, the reduced number of chains covering all terms implies a reduced number of clients in the aggregated matrix and the problem of finding the smallest n' is reduced to finding the minimum number of chains that cover all terms of the polynomial (or all vertices of the corresponding Hasse diagram). According to the well-known Dilworth's decomposition theorem (see e.g. Theorem 14.2 in [29]), this minimal number of chains is equal to the maximum size of an antichain (in our case it is the maximum number of non-embedded terms).

In order to compute the minimum number of chains we used the MINLEAF algorithm described in [30] that constructs a minimum leaf outbranching. (MINLEAF is a polynomial-time algorithm and is essentially based on finding the maximum cardinality matching.) Having such an outbranching it is possible to reconstruct the chains such that the number of chains is equal to the number of leaves in the outbranching. After that, an equivalent matrix, each column of which is induced by one of the obtained chains, can be restored. As in the formulation of the p-Median problem each column of the costs matrix corresponds to a client whose demand is to be satisfied, the existence of the equivalent matrix with a smaller number of columns implies that in the initial instance some clients can be aggregated.

Within the small example mentioned above (11) this procedure leads to the following. The reduced pseudo-Boolean polynomial $\mathcal{B}_C(\mathbf{y}) = 33 + 7y_1 + 2y_2 + 2y_4 + 2y_1y_2 + 8y_3y_4 + 4y_1y_4 + 11y_1y_2y_4 + 10y_1y_3y_4 + 4y_2y_3y_4$ corresponds to the following Hasse diagram:

$$\begin{array}{ccccccccccc} & & y_2 & \rightarrow & y_1y_2 & \rightarrow & y_1y_2y_4 & & & & \\ & \nearrow & & \nearrow & & \nearrow & & \searrow & & & \\ const & \rightarrow & y_1 & \rightarrow & y_1y_4 & \rightarrow & y_1y_3y_4 & \rightarrow & y_1y_2y_3y_4. & & \\ & \searrow & & \nearrow & & \nearrow & & \nearrow & & & \\ & & y_4 & \rightarrow & y_3y_4 & \rightarrow & y_2y_3y_4 & & & & \end{array} \tag{14}$$

It is easy to check that the size of the maximum antichain is 3, so all the terms of $\mathcal{B}_C(\mathbf{y})$ can be covered by three chains and the aggregated matrix has three columns. Below are the chains (each being presented as a column), permutation and differences matrices:

$$\begin{array}{lll} y_2 & y_1 & y_4 \\ y_1y_2 & y_1y_4 & y_3y_4 \\ y_1y_2y_4 & y_1y_3y_4 & y_2y_3y_4 \\ y_1y_2y_3y_4 & y_1y_2y_3y_4 & y_1y_2y_3y_4 \end{array} \tag{15}$$

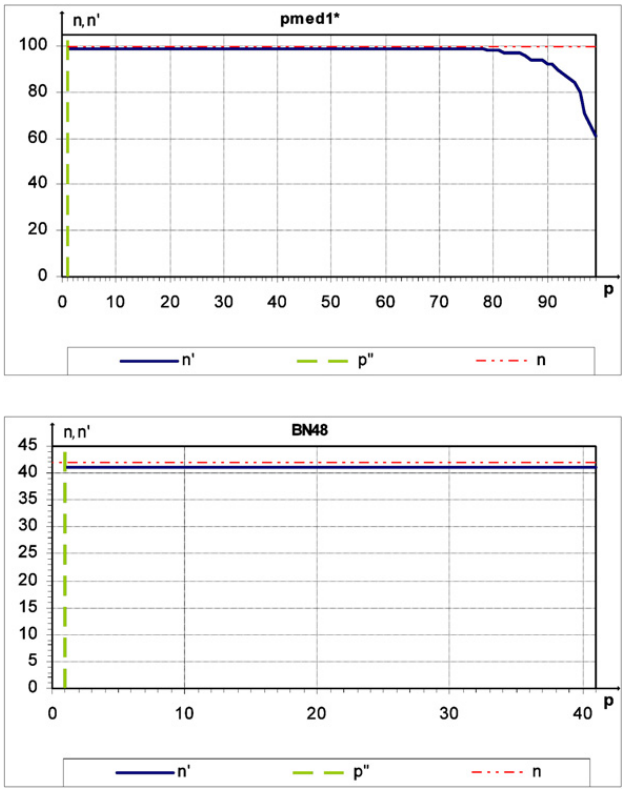

Fig. 1. Aggregation of clients for benchmark instances from OR and ODM libraries.

$$\Pi' = \begin{bmatrix} 2 & 1 & 4 \\ 1 & 4 & 3 \\ 4 & 3 & 2 \\ 3 & 2 & 1 \end{bmatrix} \qquad \Delta' = \begin{bmatrix} 0 & 0 & 33 \\ 2 & 7 & 2 \\ 2 & 4 & 8 \\ 11 & 10 & 4 \end{bmatrix}. \tag{16}$$

Having these two matrices it is possible to restore the costs matrix D of the aggregated instance:

$$D = \begin{bmatrix} 2 & 0 & 47 \\ 0 & 21 & 43 \\ 15 & 11 & 35 \\ 4 & 7 & 33 \end{bmatrix}. \tag{17}$$

4.2.1. Experiments

As was mentioned above, the minimum number of aggregated clients (columns) does not exceed n. On the other hand, it cannot be smaller then the maximum number of terms with the same degree in the reduced polynomial. In particular, for the case of instances from OR library this leads to the following. As the costs matrix for such instances has a zero diagonal, the minimal element of the ith column is located in the ith row and the first row of the permutation matrix contains no equal entries. This means that the (reduced) pBp contains n linear terms and cannot be covered by less then n chains. So, the OR instances, if considered "as is", do not allow any aggregation of clients. This result brought us to an idea of considering the corrected instances without zeroes on the diagonal (it is filled by some positive numbers during application of the Floyd's algorithm). Further we mark such instances with an asterisk (e.g. pmed1*). As all the other considered libraries are free of the mentioned "hardness", they can be directly used for experiments with the aggregation of clients.

In our experiments we considered truncated polynomials and determined the minimum number of aggregated columns (n') for all values of p from 1 to $m - 1$ (if $p = m$, the truncated polynomial is just a constant and it can be covered by one chain). Let us denote by p'' the smallest number of medians at which the truncated polynomial can be covered by less then n chains.

The results for typical representatives from each library are given in Figs. 1 and 2. As can be seen from the figures, for corrected OR and ODM problems $p'' = 0$ and even a non-truncated polynomial can be covered by $n - 1$ chains, thus making it possible to aggregate one client. At the same time, for TSP and RW instances any aggregation becomes possible only as p gets very close to m.

Thus, we can summarize that the reduction of the number of clients is negligible for all the considered benchmark libraries.

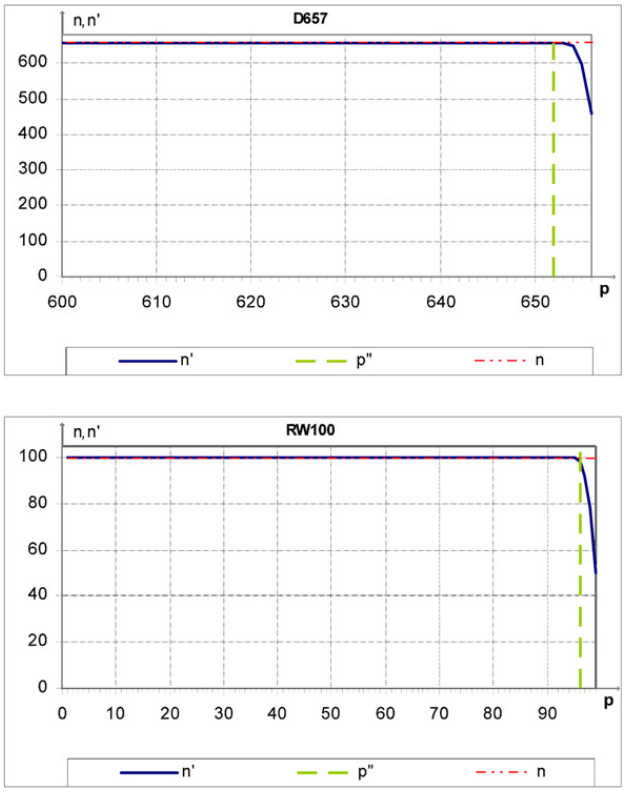

Fig. 2. Aggregation of clients for benchmark instances from TSP and RW libraries.

4.3. Minimality of the pseudo-Boolean representation

In the previous sections we described a number of reductions that are based on pseudo-Boolean formulation of the PMP and substantially reduce the amount of data that unambiguously describes the instance. However, there emerges a natural question: can one do better by using a different approach? The following lemma gives an answer to this question.

Lemma 1. *The pseudo-Boolean formulation (10) of PMP allows the most compact representation of its instance.*

Proof. Suppose the reduced and truncated pseudo-Boolean polynomial contains a monomial $\alpha\mathcal{T}$ with a nonzero coefficient that corresponds to an entry of the costs matrix c_{ij} that does not contribute to any optimal solution. This means that there exists a client j that cannot be assigned to location i. There can be several causes for this:

1. For any subset S of p opened locations there always exists a location $i' \in S$ such that $c_{i'j} < c_{ij}$. In this case client j is never served from location i.
2. For client j location i can be replaced by location i', i.e. there exists some location i' such that $c_{ij} = c_{i'j}$. In this case client j can be served from location i' instead of i.
3. For some subset of locations S client j is equivalent to some client j'. (By equivalence of clients with regard to the set of locations S we mean that sorting locations from S by distance from j and j' gives two equal sequences). In this case these two clients can be viewed as one with aggregate serving costs $c_{ij} + c_{ij'}$ for all $i \in S$.

The latter two cases are symmetric: case 2 means that from the point of view of client j locations i and i' are equally distant, while case 3 means that from the point of view of the set of locations S clients j and j' are equal. In case 1 the coefficient α is set to 0 during the truncation. For the second case we have zero coefficient as the difference $\Delta[., j] = c_{ij} - c_{i'j}$ is zero. Finally, for the third case equivalent clients are eliminated by the reduction of similar monomials. Thus, we have a contradiction and the proof is completed. □

Lemma 1 has important consequences for the applicability of pseudo-Boolean formulation. Let us consider an arbitrary model of the PMP within the class of mixed-Boolean linear programming (LP) models. The size of a mixed-Boolean LP model is determined by the following four factors:

- number of Boolean variables
- number of continuous variables
- number of constraints (and number of terms in each constraint)
- number of monomials in the objective function.

Suppose, now one tries to find the minimum model within the mentioned class. We claim that the minimum mixed-Boolean LP (MBLP) model for PMP can be derived from its pseudo-Boolean representation. Let us now consider how such

a model can be constructed. The main question here is how to transform a (nonlinear) polynomial into a linear objective function. In order to do that we introduce additional nonnegative variables $\mathbf{z}$ such that each variable z_k corresponds to a nonlinear term $\mathcal{T}_k = \prod_{i \in T_k} y_i$ of the pBp. At the same time, additional constraints are needed to represent relations between these new variables and old variables $\mathbf{y}$. If we substitute some term $\mathcal{T} = \prod_{i \in T} y_i$ by a new variable z, then the following linear constraints are sufficient:

$$\begin{aligned} &z \geq \sum_{i \in T} y_i - |T| + 1 \\ &z \geq 0. \end{aligned} \tag{18}$$

Such constraints force a z-variable to take a value of at least 1 only if all the y-variables included in $\mathcal{T}$ are set to 1. Taking into account that we have a minimization problem, z-variables will be set to 0 if at least one of the y-variables in the corresponding term is zero and to 1, otherwise. This implies that the non-negativity of the new variables is sufficient and no additional Boolean variables are introduced. The number of terms in constraints can be somewhat decreased by observing that if one has two embedded terms $\mathcal{T}_1$ and $\mathcal{T}_2$ ($T_1 \subset T_2$, i.e. $\mathcal{T}_2$ contains all variables from $\mathcal{T}_1$) and two corresponding variables z_1 and z_2 subject to:

$$\begin{aligned} z_1 &\geq \sum_{i \in T_1} y_i - |T_1| + 1 \\ z_2 &\geq \sum_{i \in T_2} y_i - |T_2| + 1 \end{aligned} \tag{19}$$

then these two inequalities can be substituted by the following two inequalities with less coefficients:

$$\begin{aligned} z_1 &\geq \sum_{i \in T_1} y_i - |T_1| + 1 \\ z_2 &\geq \sum_{i \in T_2 \setminus T_1} y_i + z_1 - |T_2 \setminus T_1|. \end{aligned} \tag{20}$$

In order to ensure the minimum possible number of coefficients in the constraints, one has to find a cover of each term from the pBp with the minimum number of its embedded terms, i.e. to solve for each term a set covering problem which is known to be NP-hard [31].

We illustrate this mixed-Boolean LP model with the following small example instance taken from [20] (see also [28]).

Example. The costs matrix

$$C = \begin{bmatrix} 1 & 6 & 5 & 3 & 4 \\ 2 & 1 & 2 & 3 & 5 \\ 1 & 2 & 3 & 3 & 3 \\ 4 & 3 & 1 & 8 & 2 \end{bmatrix}$$

leads to the following pseudo-Boolean polynomial if $p = 2$:

$$\mathcal{B}_{C,p=2} = 8 + y_2 + 2y_4 + y_1y_3 + y_2y_3 + y_2y_4 + y_3y_4.$$

By introducing new variables

$$z_5 = y_1y_3, \qquad z_6 = y_2y_3, \qquad z_7 = y_2y_4, \qquad z_8 = y_3y_4$$

and constraints

$$\begin{aligned} &z_5 \geq y_1 + y_3 - 2 + 1 \qquad z_6 \geq y_2 + y_3 - 2 + 1 \\ &z_7 \geq y_2 + y_4 - 2 + 1 \qquad z_8 \geq y_3 + y_4 - 2 + 1 \\ &z_i \geq 0, \quad i = 5, \ldots, 8 \end{aligned}$$

we have the following mixed Boolean LP model:

$$\begin{aligned} &f(\mathbf{y}, \mathbf{z}) = 8 + y_2 + 2y_4 + z_5 + z_6 + z_7 + z_8 \to \min \\ &\text{s.t.} \\ &y_1 + y_2 + y_3 + y_4 = 2 \\ &z_5 \geq y_1 + y_3 - 1 \\ &z_6 \geq y_2 + y_3 - 1 \\ &z_7 \geq y_2 + y_4 - 1 \\ &z_8 \geq y_3 + y_4 - 1 \\ &y_i \in \{0, 1\}, \quad i = 1, \ldots, 4 \\ &z_i \geq 0, \quad i = 5, \ldots, 8. \end{aligned} \tag{21}$$

The obtained model has 4 Boolean y-variables, 4 nonnegative z-variables, 5 constraints and 6 terms in the objective function. For Elloumi's Mixed Integer Linear Programming (MILP) model [20] the numbers are 4, 17, 23 and 12, respectively. In the model (21) each nonlinear monomial of the truncated and reduced pseudo-Boolean polynomial induces a linear inequality which must be added to the set of constraints. In the restrictions of (21) the new variables z_i must be just nonnegative because Boolean variables y_i imply that all z_i are Boolean. Informally, our linear constraints in (21) indicate whether or not a specific penalty to the linear part of the objective function in (21) should be added depending on the subset of opened and closed sites.

Now the pseudo-Boolean formulation of PMP with a nonlinear objective function can be presented as the following mixed Boolean linear programming model:

$$\begin{aligned}
&\left\{\alpha_0 + \sum_{r=1}^{m} \alpha_r y_r + \sum_{r=m+1}^{k} \alpha_r z_r\right\} \to \min \\
&\text{s.t.} \\
&\sum_{i=1}^{m} y_i = m - p \\
&\sum_{i \in T_r} y_i - |T_r| + 1 \le z_r, \quad r = m+1, \ldots, k \\
&y_i \in \{0, 1\}, \quad i = 1, \ldots, m \\
&z_i \ge 0, \quad i = m+1, \ldots, k.
\end{aligned} \tag{22}$$

The objective function in (22) is split into three parts: the first part α_0 is the sum of all smallest entries δ_{1j} per column (client) j — serving costs achieved under an assumption that all sites are open. The second part reflects the penalties δ_{2j} incurred by the next to the smallest entries, and the third part represents all other penalties corresponding to δ_{ij} for $1 < i \le m - p$. All other constraints are explained similarly to the example (21).

Definition 5. A *minimum MILP model* is one that minimizes the following numbers:

1. Boolean decision variables;
2. nonnegative decision variables;
3. linear constraints;
4. nonzero coefficients in the objective function.

The following theorem justifies that no improvement of the above mentioned reductions is possible.

Theorem 1. *A mixed-Boolean LP model (22) is the minimum model within the class of MILP models.*

Proof. The number of Boolean variables can be assumed fixed as one cannot use less than m such variables reflecting open and closed locations (note, that Avella and Sforza [16] use $m \times n$ Boolean variables reflecting assignment of a certain client to a certain location).

Although the original formulation of PMP contains no continuous variables, they emerge during its transformation into mixed-Boolean LP (MBLP) form. The only possible source of additional variables here is the objective function, hence we are interested in minimization of the number of terms in it.

The number of constraints is initially equal to 1, as the pseudo-Boolean formulation of PMP (10) has only one constraint requiring exactly p locations to be opened. However, additional constraints result from the objective function (e.g. during its linearization additional constraints are needed to represent relations between non-linear terms). The system of constraints in the proposed MBLP model is not redundant as all of them are linearly independent. This follows from the fact that initially any z-variable is included into exactly one constraint (18) and there is exactly one constraint that contains no z-variables — the one that requires exactly p locations to be opened.

Thus, the only space for improvement is the size of the objective function. However, due to Lemma 1, the truncated and reduced pseudo-Boolean polynomial is optimal with this respect.

As discussed in the formulation of our model, minimization of the number of coefficients in constraints is itself an NP-hard problem. This means that the number of coefficients in constraints of the proposed mixed-Boolean LP model is not minimum, while the number of constraints, number of variables and number of nonzero coefficients in the objective functions are. □

Thus, pseudo-Boolean formulation allows not only accurate estimation of the instance data complexity, but also can be used for construction of the most compact MBLP models for the PMP. As we will see in the following section, such a formulation, which we call the *Mixed-Boolean pseudo-Boolean Model* (MBpBM), also allows some further preprocessing of the instance.

5. Preprocessing

Even though the obtained MBLP formulation (21) has smaller size than other known models, it can be subjected to further reduction. We have included in this paper a reduction based on bounds (similar to the one from [11]) only with one purpose: we would like to emphasize that reductions based on bounds are out of the scope of this paper irrespectively of how efficient and useful they are.

Suppose one has computed some upper bound f^{UB} on the optimal solution. This can even be a virtual upper bound, i.e. without a feasible solution. Take now some term $\mathcal{T}_k$ from the pseudo-Boolean polynomial and define a vector $\mathbf{y}^k$ in the following way: for any $i \in T_k$ set $y_i^k = 1$ and set all other elements of $\mathbf{y}^k$ to zero. It is easy to see that $\mathcal{B}_{C,p}(\mathbf{y}^k)$ is a valid lower bound for the subspace of solutions with all locations from T_k closed. If $B_{C,p}(\mathbf{y}^k) > f^{UB}$ then, clearly, for any optimal solution at least one of the variables in term $\mathcal{T}_k$ is zero. These considerations allow one to fix some y- and z-variables to zero in the MBLP formulation and in case $f^{UB} = 9$ (computed by greedy heuristic) the example model (21) reduces to:

$$\begin{aligned}
&f(\mathbf{y}, \mathbf{z}) = 8 + y_2 + z_5 \to \min \\
&\text{s.t.} \\
&y_1 + y_2 + y_3 = 2 \\
&z_5 \geq y_1 + y_3 - 1 \\
&0 \geq y_2 + y_3 - 1 \\
&y_4 = 0 \\
&y_i \in \{0, 1\}, \quad i = 1, \ldots, 4 \\
&z_5 \geq 0.
\end{aligned} \tag{23}$$

Here are details of the computations leading to the model (23). The greedy heuristic works as follows: it starts with all locations opened (i.e. $\mathbf{y} = (0, 0, 0, 0)$) and at each step closes such a location (sets such y_i to 1) that results in the smallest increase in the value of the objective function. The procedure is repeated until $m - p$ locations are closed ($m - p$ entries of $\mathbf{y}$ are set to 1). Then, for every term $\mathcal{T}_k$ of the objective function we construct a vector $\mathbf{y}^k$ and compute $\mathcal{B}_{C,p=2}(\mathbf{y}^k)$:

$$\begin{aligned}
&\mathcal{T}_1 = y_2 \quad \mathbf{y}^1 = (0, 1, 0, 0) \quad \mathcal{B}_{C,p=2}(\mathbf{y}^1) = 9 \\
&\mathcal{T}_2 = y_4 \quad \mathbf{y}^2 = (0, 0, 0, 1) \quad \mathcal{B}_{C,p=2}(\mathbf{y}^2) = 10 > f^{UB} \\
&\mathcal{T}_3 = y_1 y_3 \quad \mathbf{y}^3 = (1, 0, 1, 0) \quad \mathcal{B}_{C,p=2}(\mathbf{y}^3) = 9 \\
&\mathcal{T}_4 = y_2 y_3 \quad \mathbf{y}^4 = (0, 1, 1, 0) \quad \mathcal{B}_{C,p=2}(\mathbf{y}^4) = 10 > f^{UB} \\
&\mathcal{T}_5 = y_2 y_4 \quad \mathbf{y}^5 = (0, 1, 0, 1) \quad \mathcal{B}_{C,p=2}(\mathbf{y}^5) = 12 > f^{UB} \\
&\mathcal{T}_6 = y_3 y_4 \quad \mathbf{y}^6 = (0, 0, 1, 1) \quad \mathcal{B}_{C,p=2}(\mathbf{y}^6) = 11 > f^{UB}.
\end{aligned} \tag{24}$$

By comparing the obtained values with the computed upper bound we have that in any optimal solution $\mathcal{T}_2$, $\mathcal{T}_4$, $\mathcal{T}_5$ and $\mathcal{T}_6$ are zero, i.e. in our MBLP model we can fix variables y_4, z_6, z_7 and z_8 to zero. By substituting the fixed values into all constraints and the objective function and observing that the third, fourth and fifth constraints in (21) become redundant we obtain the reduced model (23). Further we will call our MBLP formulation MBpBM (the mixed-Boolean pseudo-Boolean model) and its variation with reduction based on bounds – MBpBMb.

There also exist variations of this approach based on bounds (see e.g. [16]), however, such reductions crucially depend on the quality (tightness) of bounds while in this paper we defined a notion of complexity that is independent of solution algorithms. That is why we only consider here quite a restricted class of preprocessing purely based on properties of the instance data.

The essence of preprocessing that we consider is to find such locations that can be excluded from consideration as they are not contained in some optimal solution. In particular, let us define the p-truncation operation applied separately to each column of the costs matrix as setting p largest entries to the value of the smallest of them. This procedure ensures that the pBp of the p-truncated matrix is equal to the truncated pBp of the initial matrix. The following theorem, cited from [28] (see Theorem 4), provides a direct suggestion for preprocessing based on p-truncation.

Theorem 2. *Assume that in a given PMP instance, all the entries corresponding to a particular row i in the costs matrix C are changed when p-truncation operations are performed on all columns of C. Then there exists an optimal solution $\mathbf{y}^*$ to the instance with $y_i^* = 1$.*

In other words, the theorem means that if some variable y_i is not contained in the truncated polynomial, then there exists an optimal solution $\mathbf{y}^*$ with $y_i^* = 1$. In order to illustrate this we would like to consider the following example. Let the costs matrix be defined as (the rightmost column of numbers enumerates rows of the matrix):

$$C = \begin{bmatrix} 1 & 3 & 9 \\ 2 & 5 & 3 \\ 9 & 7 & 8 \\ 5 & 9 & 7 \\ 4 & 4 & 5 \end{bmatrix} \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5. \end{matrix} \tag{25}$$

Table 2
Values of p' and p^* for benchmark instances.

Library	Instance	m	p'	p^*
OR	pmed1*	100	90	93
OR	pmed15*	300	180	285
OR	pmed26*	600	452	581
OR	pmed40*	900	644	882
ODM	BN48	42	27	35
ODM	BN1284	1284	653	1211
ODM	BN3773	3773	3385	3742
ODM	BN5535	5535	2179	5503
TSP	rd100	100	97	97
TSP	D657	657	477	653
TSP	fl1400	1400	1177	1395
TSP	pcb3038	3038	3026	3033
RW	rw100	100	90	95
RW	rw200	200	186	193
RW	rw250	250	241	243
RW	rw500	500	489	492
RW	rw1000	1000	978	992

Also, let p be $p = \lceil m/2 \rceil = 3$ (which corresponds to the hardest case from a combinatorial point of view). The p-truncated matrix $C_{p=3}$ is:

$$C_{p=3} = \begin{bmatrix} 1 & 3 & 7 \\ 2 & 5 & 3 \\ 4 & 5 & 7 \\ 4 & 5 & 7 \\ 4 & 4 & 5 \end{bmatrix} \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5. \end{matrix} \tag{26}$$

The objective function can be represented by the pseudo-Boolean polynomial $\mathcal{B}_C(\mathbf{y}) = 7 + 2y_1 + 2y_2 + 2y_1y_2 + 1y_1y_5 + 2y_2y_5 + 3y_1y_2y_5 + 1y_2y_4y_5 + 4y_1y_2y_4y_5 + 2y_1y_2y_3y_5 + 1y_2y_3y_4y_5$. After truncation one obtains $\mathcal{B}_{C,p=3}(\mathbf{y}) = 7 + 2y_1 + 2y_2 + 2y_1y_2 + 1y_1y_5 + 2y_2y_5$. As can be seen, the truncated pBp does not contain two variables y_3, y_4, so they can be set to 1 as this does not affect the value of $\mathcal{B}_{C,p=3}(\mathbf{y})$. This means that the initial matrix C given by (25) can be reduced to matrix D with fewer rows:

$$D = \begin{bmatrix} 1 & 3 & 7 \\ 2 & 5 & 3 \\ 4 & 4 & 5 \end{bmatrix} \begin{matrix} 1 \\ 2 \\ 5. \end{matrix} \tag{27}$$

It should be noticed that if one sets all other variables y_1, y_2, y_5 to 0, this immediately gives the optimal solution. Thus, for this small example the problem can be solved just by data preprocessing.

However, with large instances this technique does not always allow one to solve the problem. Given a PMP costs matrix, we studied how the possibility of preprocessing depends on the value of p. As the value of p grows, the number of entries in any column whose values are revised increases. So, the higher the value of p, the greater the chance that a row of C is eliminated due to Theorem 2. This explains why PMP instances with $p = p_0$, $p_0 < m/2$, are more difficult to solve than instances on the same costs matrix with $p = m - p_0$, even though the number of feasible solutions for both cases are identical. Let p' be the smallest value of p for which p-truncation eliminates at least one row in C. Let us also denote by p^* the minimum number of rows that contain the minimum entry of each column of C. Then, the PMP instance defined on C with $p > p^*$ has open facilities that do not serve any client. Hence, for $p_1 > p^*$, the number of optimal solutions has a lower bound of $\binom{m-p^*}{p_1-p^*}$. So, for PMP instances with $p > p^*$ it becomes progressively more difficult to prove the optimality of a solution as the value of p increases from p^* to $(m-p^*)/2$. However, this becomes progressively easier as p increases further. Table 2 presents a characterization of benchmark instances introduced in Table 1 in terms of p' and p^*.

6. Complex benchmark instances

In this section we consider the aspects of constructing complex benchmark instances that can be used for testing solution algorithms and introduce our library of such instances.

To have maximum possible complexity, a PMP instance defined on an $m \times n$ costs matrix should not be amenable to any of the reductions described above. Thus, first of all, the entries of the differences matrix should be non-zero, such that all monomials in the pBp have non-zero coefficients, or, equivalently:

Claim 1. *The most complex instances have pairwise different and nonzero entries in every column of the costs matrix (assuming that the sizes of the costs matrix are fixed).*

However, as explained below, these two restrictions on the entries of the differences matrix are not sufficient to ensure the complexity. Suppose, for some column j the difference between the minimal and second minimal element $\Delta c[1, j]$ is comparable to the (unknown) costs of the optimal solution. In this case the location (row), at which the minimum for jth client (column) is attained, will be included into any optimal solution. Such additional structure can be exploited by the solution algorithms and thus reduces the complexity of the instance. This particular case can be generalized in the following way. Suppose, the (truncated) pseudo-Boolean polynomial contains a monomial $\alpha\mathcal{T} = \alpha\prod_{i\in T} y_i$ with a large enough coefficient α that exceeds the costs of the optimal solution (or its somehow computed upper bound). Then, clearly, for any optimal solution $\mathbf{y}$ holds $\mathcal{T}(\mathbf{y}) = 0$, implying that at least one of the variables in $\mathcal{T}$ must be set to zero and at least one of the corresponding locations is opened. In fact, this condition can be made even stronger if one considers not only the coefficient α at $\mathcal{T}$, but the sum of α and the coefficients of all monomials with terms embedded in $\mathcal{T}$. It is also quite straightforward that this test is more likely to fail as the range of the entries of the differences matrix (or, equivalently, coefficients of the pseudo-Boolean polynomial) becomes smaller, up to the limit case when they are all equal. These considerations lead to the following claim.

Claim 2. *Instances that lead to the pseudo-Boolean polynomial with all coefficients equal (except a constant — monomial of degree zero) are the most complex ones (assuming that the number of monomials is fixed).*

Once we know how to construct a "complex" pseudo-Boolean polynomial, we are interested in maximizing the number of monomials in it. To achieve this, there should be no similar monomials in the pBp representation of the problem. It should be mentioned that constants obtained from pseudo-Boolean representation of all the columns can be reduced into one monomial, so every PMP instance has a complexity of at most

$$comp(\mathcal{C}) \leq mn - (n-1) = n(m-1) + 1. \tag{28}$$

To ensure that only constants can be aggregated, the permutation matrix Π must conform with the following requirement: the sets of the first k entries of columns Π^j in Π should be pairwise different for any $k : 1 \leq k \leq m$. This requirement can be expressed in an alternative form. Let us consider a Hasse diagram defined over the subsets of $\{1, \ldots, m\}$. It is easy to see that each permutation $\Pi^j = (\pi_{1j}, \pi_{2j}, \ldots, \pi_{mj})^{\mathrm{T}}$ corresponds to a chain of embedded subsets $\{\pi_{1j}\} \subset \{\pi_{1j}, \pi_{2j}\} \subset \cdots \subset \{\pi_{1j}, \ldots, \pi_{mj}\}$ that, in turn, corresponds to a $\emptyset - \{1, \ldots, m\}$ path in the Hasse diagram. Now the requirement can be formulated as follows.

Claim 3. *In order to prohibit the reduction of similar monomials, the permutation matrix should correspond to a collection of internally vertex-disjoint $\emptyset - \{1, \ldots, m\}$ paths in the Hasse diagram defined on subsets of $\{1, \ldots, m\}$.*

Taking into account that there are at most m such paths, for PMP instances with $n > m$ it is always possible to reduce at least $n - m$ linear monomials in the pBp, so for instances in our benchmark library holds $n \leq m$. Due to these considerations it is possible to formulate the problem of constructing a permutation matrix that leads to a complex instance as a problem of finding n vertex disjoint paths in a graph obtained from the Hasse diagram. Though this problem is known to be polynomially solvable [32], the fact that the complete Hasse diagram has 2^m vertices makes the procedure very time consuming for large m. However, there exists a trivial solution:

$$\pi_{ij} = (i + j) \bmod m + 1. \tag{29}$$

In case $n = 4, m = 5$ this solution leads to the following permutation matrix Π:

$$\Pi = \begin{bmatrix} 3 & 4 & 5 & 1 \\ 4 & 5 & 1 & 2 \\ 5 & 1 & 2 & 3 \\ 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 5 \end{bmatrix}.$$

Based on a representation of the monomials as a collection of chains it is possible to estimate the complexity of a PMP instance (the maximum number of monomials in the pBp). Consider a complete Hasse diagram that contains all subsets of $\{1, \ldots, m\}$. Clearly, the maximum length of a chain of embedded non-constant terms is $m-1$, as it is the maximum possible degree of the pBp. The number of chains of this maximum length is exactly $m = \binom{m}{1}$ as there exist m linear terms (as well as m terms of degree $m-1$). Each of these chains uses exactly one term of each degree from 1 to $m-1$. Once all maximum length chains are used, the next available maximum length of a chain is $m-3$ (terms of degree $2, \ldots, m-2$). The number of such chains is $\binom{m}{2} - \binom{m}{1}$ which is exactly the number of quadratic terms that were not included in chains of length $m-1$. If we have enough columns to use all these chains (i.e. n is sufficiently large) then we switch to chains of length $m-5$ (terms of degree $3, \ldots, m-3$) and there are $\binom{m}{3} - [\binom{m}{2} - \binom{m}{1}] - \binom{m}{1} = \binom{m}{3} - \binom{m}{2}$ such chains, which is exactly the number of cubic terms not used by chains of lengths $m-2$ and $m-1$. We continue picking the longest possible chains until we have n of them. It is not hard to understand that the number of terms of some degree k ($1 \leq k \leq m-1$) in such a collection of n longest chains is bounded by n and, at the same time, cannot exceed $\binom{m}{k}$. Fig. 3 gives a graphical representation of how the number of terms in such a collection of chains of maximum length can be calculated.

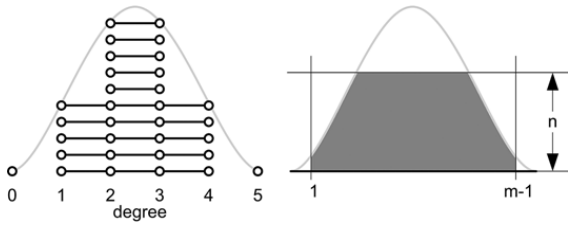

Fig. 3. Estimating the maximum number of nonzero terms in a pBp.

In the left part of Fig. 3 an example for $m = 5$ is shown. Circles denote terms of a pBp that are arranged in such a way that terms of the same degree are within one column. Links correspond to possible chains of embedded terms. If one is aimed at having n chains containing the maximum number of terms then he picks n longest chains starting from the lower part of the picture. In particular, it can be seen that it is impossible to get more than 5 full chains. For example, if $n = 6$ at least one linear monomial will be reduced. For arbitrary m and n the maximum number of monomials in the reduced pBp corresponds to the area of the shaded region in the right part of Fig. 3. Thus the complexity (equivalently, the number of monomials in the corresponding pBp) of a PMP instance $\mathcal{C}$ defined by an $m \times n$ costs matrix is bounded by

$$comp(\mathcal{C}) \leq \sum_{i=1}^{m-1} \min\left\{n, \binom{m}{i}\right\} + 1. \tag{30}$$

The main peculiarity here is that for a number of clients n exceeding the number of locations m the addition of new clients has a progressively smaller impact on the complexity of the instance that is always less than $n(m-1)+1$, while for $n \leq m$ there exist instances of complexity $n(m-1)+1$.

It is easy to see that in case $n < m$ all the minima are contained in at most n rows (i.e. all minima are achieved on at most $n < m$ locations) and preprocessing will eliminate at least $m - n$ variables (rows of the costs matrix). This leads to a conclusion that instances with $n = m$ are, potentially, the most complex ones (provided the entries of the costs matrix satisfy the requirements considered above).

Possibility of truncation of the pseudo-Boolean polynomial depends only on the number of medians and is not affected by the values of the costs matrix. Thus, we cannot negate this reduction by adjustment of the costs matrix and if p is fixed (30) can be improved in the following way:

$$comp(\mathcal{C}) \leq \sum_{i=1}^{m-p} \min\left\{n, \binom{m}{i}\right\} + 1. \tag{31}$$

Our benchmark library contains complex (in terms of the possibility of problem size reduction) PMP instances defined on square matrices of different size. As costs matrices are dense, they are stored explicitly in files named "Xmatr$Y$, Z.txt", where X is 't' if the permutation matrix Π is defined by (29) and X is 'r' if Π is a randomized permutation matrix obtained as a solution to the disjoint paths problem mentioned above. Y reflects the values of m and n (in our instances $n = m$), and costs are selected in such a way that entries of the differences matrix Δ are uniformly distributed random integers from $\{1, \ldots, Z\}$. Due to Claim 2 instances with smaller Z are harder to solve. For example, the file named "tmatr4,1.txt" defines the following instance:

$$C = \begin{bmatrix} 3 & 4 & 1 & 2 \\ 4 & 1 & 2 & 3 \\ 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 1 \end{bmatrix}. \tag{32}$$

It is easy to check that the permutation matrix is the same as costs matrix C and the difference matrix has all unit entries.

The structure of the files is as follows. The first line contains the numbers of clients and potential locations (columns and rows of the costs matrix). Next all entries of the costs matrix are explicitly listed row by row. The library is available at http://www.hse.ru/en/org/persons/22927115.

We would like to finish the description of our complex instances by mentioning that under certain conditions optimal objective values can be computed by a simple formula (see Lemma 2 below). This property is especially useful for the developers of heuristic methods and makes it possible to estimate the quality of the generated solutions.

Lemma 2. *If the $m \times n$ costs matrix of a PMP instance satisfies the following conditions:*

(i) $m = n$,
(ii) *a permutation matrix is defined by (29),*
(iii) *all entries of the differences matrix are equal to some constant d,*

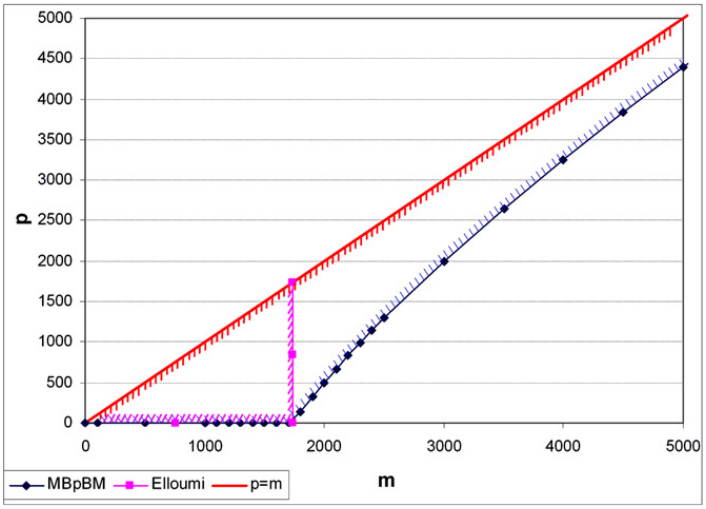

Fig. 4. Ranges of complex instance data size for which our MBpBM and Elloumi's NF can be loaded by Xpress.

then the optimal objective value can be computed as:

$$d(n'+1)\left[\frac{n'p}{2}+(n \bmod p)\right], \tag{33}$$

where $n' = \lfloor n/p \rfloor$.

Proof. The conditions of the lemma ensure that each row (and each column) of the costs matrix can be obtained from $(d, 2d, \ldots, md)$ by cyclic shift, i.e. each multiple of d is contained in each row exactly once. This implies that at most p clients can be served at a cost d. Also, at most p clients can be served at costs $2d$, $3d$, etc. Thus, the minimum can be obtained by serving first p clients at a cost d, next $\min\{n-p, p\}$ clients at a cost $2d$, next $\min\{n-2p, p\}$ clients at a cost $3d$, etc., until we serve all n clients. By a simple combinatorial reasoning, the total costs in this case can be computed as

$$d\left[\frac{n'(n'+1)}{2}p+(n'+1)(n \bmod p)\right] = d(n'+1)\left[\frac{n'p}{2}+(n \bmod p)\right]. \tag{34}$$

This minimum is achieved by the p locations (rows of the costs matrix) that fall within the following pattern:

$$\begin{array}{cccccccccccc} (d & 2d & \ldots & pd & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & md) \\ (\ldots & \ldots & \ldots & md & d & 2d & \ldots & pd & \ldots & \ldots & \ldots) \\ (\ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & md & d & 2d & \ldots) \\ \ldots. \end{array} \quad \square \tag{35}$$

Lemma 2 and its constructive proof have an important corollary. It can be checked that in the instances satisfying the condition of the lemma each location is open in p optimal solutions and the number of optimal solutions is n (does not depend on the number of medians p). This means that these instances are degenerate and may be easily solvable, irrespectively of their size.

In order to check the properties of our instances we held a number of computational experiments. For the sake of comparison we used two formulations of PMP: our MBpBM and Elloumi's NF [20] (which is the most compact MILP formulation of PMP, to the best of our knowledge). Fig. 4 shows the ranges of m and p for which the model can be loaded into the MILP solver (in our case Xpress). For different sizes of the $m \times m$ input matrix we checked for which range of p the formulation can be loaded into the MILP solver (i.e., is small enough to fit into the memory). Clearly, this range is bounded from above by the line $p = m$. As Elloumi's formulation does not account for the number of medians, there exists some critical size of the cost matrix beyond which the formulation becomes prohibitively large, irrespectively of p. At the same time, our formulation based on the pseudo-Boolean representation of the instance data can be loaded by a general-purpose MILP solver for some values of p even if the input matrix is of huge dimension (see Fig. 4).

Finally, we compared the running times of two solution approaches (our MBpBM and Elloumi's NF) applied to selected OR instances and to our generated instances of the same size and with the same number of medians p. As was presumed, instances with permutations given by (29) are easy for the MILP solver and the running times are of the same magnitude as running times for OR instances (even though the number of coefficients in the formulation is much larger). Thus, we compared the running times of OR instances and our complex instances with randomized permutation matrices and differences matrices containing all unit entries. The results of this comparison are presented in Table 3 and show that our benchmark instances are also complex in terms of running time. In particular, for small values of p computation times

Table 3
Running times in seconds for our MBpBM and Elloumi's NF for OR instances and our complex instances of the corresponding size.

m	p	OR instances			Our instances		
		MBpBM	MBpBMb	Elloumi	MBpBM	MBpBMb	Elloumi
100	5	0.22	0.20	0.25	4434.66	3443.13	24684.42
	10	1.47	0.58	4.08	878.78	1141.05	3926.20
	20	0.11	0.06	0.14	92.95	26.25	62.94
	33	0.22	0.05	0.13	0.28	0.11	0.61
200	5	15.22	17.67	17.06	[a]	[a]	[a]
	10	0.73	0.55	0.77	[a]	[a]	[a]
	20	0.49	0.31	0.55	[a]	[a]	[a]
	40	0.41	0.28	0.45	1616.33	1218.45	1753.47
	67	0.27	0.14	0.41	1.08	0.63	1.34
300	5	4.00	4.61	4.50	[a]	[a]	[a]
	10	8.59	8.33	7.36	[a]	[a]	[a]
	30	0.80	0.56	1.25	[a]	[a]	[a]
	60	1.05	1.13	2.34	[a]	[a]	[a]
	100	0.48	0.30	0.86	1.16	0.27	1.81
400	5	42.47	30.78	23.38	[a]	[a]	[a]
	10	25.16	21.19	32.02	[a]	[a]	[a]
	40	1.73	1.31	2.97	[a]	[a]	[a]
	80	0.97	0.72	1.61	[a]	[a]	[a]
	133	0.73	0.80	1.25	3.83	1.86	6.28
500	5	4.52	3.92	6.22	[a]	[a]	[a]
	10	51.63	64.05	98.59	[a]	[a]	[a]
	50	1.74	1.31	2.77	[a]	[a]	[a]
	100	1.42	0.97	2.33	[a]	[a]	[a]
	167	1.44	0.88	1.84	14.91	4.14	18.56
600	5	163.84	111.81	180.31	[a]	[a]	[a]
	10	27.59	21.31	43.73	[a]	[a]	[a]
	60	2.48	2.13	3.61	[a]	[a]	[a]
	120	1.78	1.31	2.91	[a]	[a]	[a]
	200	1.50	0.78	4.81	49.81	15.41	201.39
700	5	153.22	57.05	90.95	[a]	[a]	[a]
	10	33.13	43.39	37.64	[a]	[a]	[a]
	70	3.09	2.69	4.73	[a]	[a]	[a]
	140	3.72	1.97	7.11	[a]	[a]	[a]
800	5	70.30	154.41	514.72	[a]	[a]	[a]
	10	2256.83	4252.13	6737.25	[a]	[a]	[a]
	80	3.91	3.08	7.00	[a]	[a]	[a]
900	5	1328.34	2041.28	1143.97	[a]	[a]	[a]
	10	572.81	444.08	473.95	[a]	[a]	[a]
	90	5.39	4.02	8.42	[a]	[a]	[a]

[a] Not solved within 24 h.

explode even for 100×100 input matrices. Also, for the unsolved instances we compared the best found integer solutions with solutions obtained by heuristics and it was observed that heuristics produced better solutions. This contradicts the common observation that MILP solvers based on branch-and-bound procedures spend only a very small portion of the total running time on finding the optimal solution (while most of the time is spent on proving its optimality). Thus, from this point of view, our instances with randomized permutation matrices are also complex.

7. Summary and future research

Initially, the space required to represent an instance of the p-Median problem (PMP) is the size of the costs matrix of this instance. However, as shown in [28], it is always possible to represent such an instance in a more compact manner and according to our computational experiments this reduction can be quite substantial. This implies that the size of the stored PMP instances is always less than mn. In this paper we introduced a notion of data instance complexity and presented several techniques for estimating this complexity by the construction of reduced instances. We have used a formulation of the PMP in terms of a pseudo-Boolean polynomial that we call the truncated Hammer–Beresnev polynomial. Based on this formulation we developed several size reduction techniques.

The first reduction is directly derived from the polynomial representation of the problem and excludes all equal entries per column in the original instance. The second one is achieved by reducing similar monomials in the pBp and aggregates columns entries with the same monotonicity w.r.t. the original numbering of rows in the costs matrix. According to our computational experiments, these techniques lead to a significant reduction for OR, TSP and RW instances, although with ODM instances the effect is minor. The third reduction is based on truncation of the polynomial from degree $(m - 1)$ to $(m - p)$. The fourth technique is aimed rather at dense representation of the problem than at reduction of its size itself, since it does not affect the number of monomials in the pBp. It suggests aggregation of the clients (or columns of the costs

matrix). Computational experiments show that the first of the latter two techniques works equally well for all libraries (e.g. if $p = m/2$ and the number of feasible solutions is the largest, truncation leads to halved size of the stored instance), while the second one has a negligible effect.

We also showed that our reductions can be used to construct the minimum mixed-Boolean LP model for the p-Median problem.

In Section 5 we presented a preprocessing procedure for the p-Median problem based on truncated Hammer–Beresnev polynomial formulation. This formulation allows p-truncation of the costs matrix of the instance, which, in turn, allows exclusion of certain facilities from consideration as they are guaranteed not to belong to an optimal solution. If combined, this p-truncation and aggregation of clients (described in Section 4) allows reduction of both sizes of the costs matrix (numbers of columns and rows). Moreover, in terms of truncated polynomials it becomes quite straightforward why PMP problems with p_1 close to m are easier than those with small $p_2 = m - p_1$, even though the number of feasible solutions is the same.

Summarizing the experimental results, we claim that OR instances (which are most commonly used in the literature) have the least complexity while ODM instances are the most complex ones, compared to the other considered libraries, as they allow the smallest reduction. Thus, the ODM library provides the most reliable data for testing exact and heuristic PMP solving algorithms.

In Section 6 we examined the properties of the complex instances and introduced our benchmark library of PMP instances. Some of our benchmark instances (those with a permutation matrix given by (29)) have the following attractive properties: (i) the optimal value and an optimal solution are easily (polynomially) computable (see Lemma 2) and, hence, useful for the quality evaluation of both exact and heuristic algorithms for PMP; (ii) the set of optimal solutions is of large cardinality; hence, the corresponding MILP models are highly degenerate. These instances should be easy for the solution algorithms. On the contrary, our benchmark instances with randomized permutation matrices are expected to be very difficult for any exact solution algorithm.

Taking into account that pseudo-Boolean representation allows the highest problem size reduction (comparatively to other representations of the PMP), the obtained results suggest the following direction of future research: the design of new exact and heuristic algorithms for solving large-scale PMP instances based on the truncated Hammer–Beresnev polynomial and the preprocessing scheme demonstrated in Section 5.

Acknowledgements

Both authors are thankful to the Editor and Reviewers for the constructive suggestions which led to an improved presentation of this paper. The financial support from The Higher School of Economics within the project 11-04-0008 "Calculus of tolerances in combinatorial optimization: theory and algorithms" is gratefully acknowledged by the first author.

References

[1] J. Reese, Solution methods for the p-Median problem: an annotated bibliography, Networks 48 (3) (2006) 125–142.
[2] N. Mladenovic, J. Brimberg, P. Hansen, J.A. Moreno-Peréz, The p-Median problem: a survey of metaheuristic approaches, Eur. J. Oper. Res. 179 (2007) 927–939.
[3] O. Kariv, L. Hakimi, An algorithmic approach to network location problems, part II: the p-Medians, SIAM J. Appl. Math. 37 (3) (1979) 539–560.
[4] C.S. Revelle, H.A. Eiselt, M.S. Daskin, A bibliography for some fundamental problem categories in discrete location science, Eur. J. Oper. Res. 184 (2008) 817–848.
[5] B. Mirkin, Clustering For Data Mining: A Data Recovery Approach (Chapman & Hall/Crc Computer Science), Chapman & Hall/CRC, 2005.
[6] P. Avella, A. Sassano, I. Vasil'ev, Computational study of large-scale p-Median problems, Math. Program. A 109 (2007) 89–114.
[7] O.R. Library, Available at the web address http://mscmga.ms.ic.ac.uk/info.html.
[8] O. Briant, D. Naddef, The optimal diversity management problem, Oper. Res. 52 (4) (2004) 515–526.
[9] E.D. Andersen, K.D. Andersen, Presolving in linear programming, Math. Program. 71 (1995) 221–245.
[10] H. Crowder, E. Johnson, M.W. Padberg, Solving large-scale zero one linear programming problems, Oper. Res. 31 (1983) 803–834.
[11] B. Goldengorin, G.A. Tijssen, D. Ghosh, G. Sierksma, Solving the simple plant location problems using a data correcting approach, J. Global Optim. 25 (2003) 377–406.
[12] K.L. Hoffman, M.W. Padberg, Improved LP-representations of zero-one linear programs for branch-and-cut, ORSA J. Comput. 3 (1991) 121–134.
[13] A. Martin, Integer Programs with Block Structure, Technische Universität Berlin, Habilitations-Schrift, Berlin, Germany, 1998.
[14] A. Martin, General mixed integer programming: computational issues for branch-and-cut algorithms, Lect. Notes Comput. Sc. 2241 (2001) 1–25.
[15] U.H. Suhl, R. Szymanski, Super node processing of mixed-integer models, Comput. Optim. Appl. 3 (1994) 317–331.
[16] P. Avella, A. Sforza, Logical reduction tests for the p-Median problem, Ann. Oper. Res. 86 (1999) 105–115.
[17] P.L. Hammer, Plant location — a pseudo-Boolean approach, Israel J. Technol. 6 (1968) 330–332.
[18] V.L. Beresnev, On a problem of mathematical standardization theory, Upravliajemyje Sistemy 11 (1973) 43–54. in Russian.
[19] G. Cornuejols, G. Nemhauser, L.A. Wolsey, A canonical representation of simple plant location problems and its applications, SIAM J. Alg. Disc. Meth. 1 (3) (1980) 261–272.
[20] S. Elloumi, A tighter formulation of the p-Median problem, J. Comb. Optim. 19 (2010) 69–83.
[21] R.L. Church, COBRA: A new formulation of the classic p-Median location problem, Ann. Oper. Res. 122 (2003) 103–120.
[22] P. Dearing, P.L. Hammer, B. Simeone, Boolean and graph theoretic formulations of the simple plant location problem, Transport. Sci. 26 (2) (1992) 138–148.
[23] J.E. Beasley, A note on solving large p-Median problems, Eur. J. Oper. Res. 21 (1985) 270–273.
[24] G. Reinelt, TSPLIB: a traveling salesman problem library, ORSA J. Comput. 3 (1991) 376–384. http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/.
[25] M.G.C. Resende, R.F. Werneck, On the implementation of a swap-based local search procedure for the p-Median problem, in: R.E. Ladner (Ed.), Proceedings of the Fifth Workshop on Algorithm Engineering and Experiments, ALENEX'03, SIAM, 2003, pp. 119–127.
[26] C.S. ReVelle, R. Swain, Central facilities location, Geogr. Anal. 2 (1970) 30–42.
[27] P. Avella, A. Sassano, On the p-Median polytope, Math. Program. 89 (2001) 395–411.

[28] B.F. AlBdaiwi, D. Ghosh, B. Goldengorin, Data aggregation for p-Median problems, J. Comb. Optim. 21 (2011) doi:10.1007/s10878-009-9251-8 (in press).
[29] A. Schrijver, Combinatorial Optimization. Polyhedra and Efficiency, Springer, 2003, p. 218.
[30] G. Gutin, I. Razgon, E.J. Kim, Minimum leaf out-branching and related problems, in: Proc. AAIM'08, in: Lect. Notes Comput. Sc., vol. 5034, 2008, pp. 235–246.
[31] M. Garey, D. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W.H. Freeman, 1979.
[32] N. Robertson, P.D. Seymour, Graph minors. XIII. The disjoint path problem, J. Comb. Theory B 63 (1995) 65–110. doi:10.1016/j.physletb.2003.10.071.