Computers & Operations Research 39 (2012) 1625–1632

Contents lists available at SciVerse ScienceDirect

Computers & Operations Research

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

An aggregation heuristic for large scale *p*-median problem

Pasquale Avella [a], Maurizio Boccia [a], Saverio Salerno [b], Igor Vasilyev [c,*]

[a] Dipartimento di Ingegneria, Università del Sannio, Viale Traiano, 82100 Benevento, Italy
[b] Dipartimento di Ingegneria dell'Informazione e Matematica Applicata, Università di Salerno, via Ponte don Melillo, 84084 Fisciano (SA), Italy
[c] Institute of System Dynamics and Control Theory, Siberian Branch of Russian Academy of Sciences, Lermontov Srt., 134, 664033 Irkutsk, Russia

ARTICLE INFO

Available online 28 September 2011

Keywords:
p-Median problem
Clustering analysis
Lagrangean relaxation
Core heuristic
Aggregation procedure

ABSTRACT

The *p*-median problem (PMP) consists of locating *p* facilities (medians) in order to minimize the sum of distances from each client to the nearest facility. The interest in the large-scale PMP arises from applications in cluster analysis, where a set of patterns has to be partitioned into subsets (clusters) on the base of similarity.

In this paper we introduce a new heuristic for large-scale PMP instances, based on Lagrangean relaxation. It consists of three main components: subgradient column generation, combining subgradient optimization with column generation; a "core" heuristic, which computes an upper bound by solving a reduced problem defined by a subset of the original variables chosen on a base of Lagrangean reduced costs; and an aggregation procedure that defines reduced size instances by aggregating together clients with the facilities. Computational results show that the proposed heuristic is able to compute good quality lower and upper bounds for instances up to 90,000 clients and potential facilities.

1. Introduction

Given a set $I=\{1,\ldots,m\}$ of potential locations of p facilities, a set $J=\{1,\ldots,n\}$ of clients, and d_{ij} ($d(i,j)$)—given distances (transportation costs) between the location i and the client j. The p-median problem (PMP) consists of locating p facilities (medians) at locations of I in order to minimize the sum of distances from each client to the nearest facility. The combinatorial optimization formulation of PMP takes the following form:

$$Z^*=\min_{T\subset I}\left\{\sum_{j\in J}\min_{t\in T} d_{tj}:\ |T|=p\right\}.$$

The problem was introduced by Hakimi [11,12] and is known to be NP-hard [21] (see [24] for a more general survey on discrete location problems). In many applications we have $I=J$ and we can define the problem on the weighted directed graph $G(I,A)$, where I is the vertex set, A is the arc set, and weights d_{ij} are associated with the arcs $ij\in A$.

Let y_i be a binary variable which is 1 if i is a median, 0 otherwise, and x_{ij} a binary variable which is 1 if the median i is nearest from the vertex j, 0 otherwise. Let also $\delta^-(j)$ be the set of the arcs entering the vertex j. Then a mixed integer programming (MIP) model of PMP over the graph $G(I,A)$ is

$$Z^*=\min_{(x,y)}\ \sum_{ij\in A} d_{ij}x_{ij}, \tag{1}$$

$$\sum_{i\in\delta^-(j)} x_{ij}+y_j=1\quad \forall j\in I, \tag{2}$$

$$x_{ij}\le y_i\quad \forall ij\in A, \tag{3}$$

$$\sum_{i\in I} y_i=p, \tag{4}$$

$$y_i\in\{0,1\}\quad \forall i\in I, \tag{5}$$

$$x_{ij}\in\{0,1\}\quad \forall ij\in A. \tag{6}$$

Constraints (2) ensure that either j is a median or it must be assigned to a median. Variable upper bound (VUB) constraints (3) impose that a vertex can only be assigned to medians. Constraint (4) enforces the number of medians to be p. A feasible solution of the problem consists of p "stars" where medians have leaving arcs as shown in Fig. 1.

An interesting application of large-scale PMP arises in cluster analysis [33,28,26,15,32,14]. Cluster analysis consists of partitioning a set of patterns into subsets (clusters) based on similarity, i.e. a cluster has to contain the similar patterns and dissimilar patterns have to be in different clusters.

Each pattern is usually expressed by a multidimensional vector, called "feature vector", and the dissimilarity between two patterns is measured as the distance between the two

* Corresponding author. Tel.: +7 9148752836; fax: +7 3952511616.
E-mail addresses: avella@unisannio.it (P. Avella), maurizio.boccia@unisannio.it (M. Boccia), salerno@unisa.it (S. Salerno), vil@icc.ru (I. Vasilyev).

0305-0548/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2011.09.016

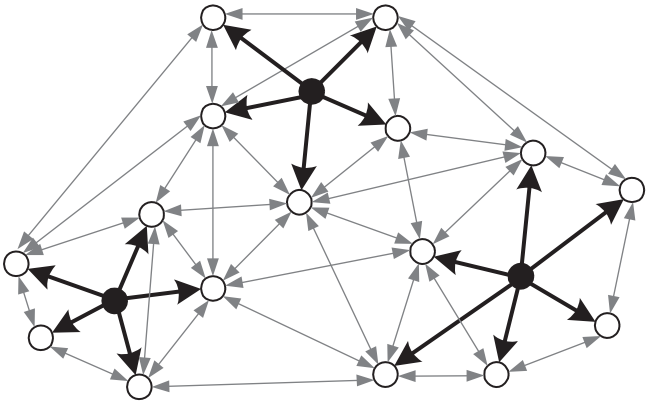

Fig. 1. A feasible solution of the PMP.

associated feature vectors, computed according to a given metric. A comprehensive review of main aspects of cluster analysis can be found in the surveys of Hansen and Jaumard [15], Jain et al. [20]. In terms of cluster analysis, the set of the patterns to be clustered corresponds to the set of the vertices of the complete digraph $G(I,A)$, with arc weights expressing the similarity between each pair of patterns. The "stars" of the PMP solution are the desired clusters. Klastorin [22] and Hansen et al. [14] showed that the p-median formulation is effective in providing good clusterings.

PMP has been widely addressed in the literature. A survey of state-of-the-art exact and heuristic algorithms can be found in Mladenović et al. [25], Avella et al. [2] and Garcia et al. [10]. Notwithstanding the fact that the problem is NP-hard, Kochetov and Ivanenko [23] showed that "metric" instances (i.e. those where distances satisfy the triangle inequality as in the cluster analysis) show small duality gaps and many heuristics proved to be effective in providing good upper bounds (the variable neighborhood search of Hansen et al. [16], the hybrid methods of Taillard [32], GRASP heuristic of Resende and Werneck [29]). Some more recent progress in the location problems and their applications can be found in Refs. [13,18,27].

Good lower bounds were obtained using Lagrangean relaxations (Beasley [3], Lagrangean/surrogate relaxation of Senna et al. [31,30], semi-Lagrangean relaxation of Beltrán et al. [4]) on the instances with a few thousand vertices. But all of them suffer from the size limitations mainly posed by the memory limit and input/output requirements.

Recently, Hansen et al. [14] presented a primal–dual variable neighborhood search (PDVNS) algorithm, providing lower bounds for instances up to 20,000 vertices and upper bounds for instances up to 90,000 vertices, using a standard workstation.

In this paper we introduce a new heuristic approach based on Lagrangean relaxation which consists of the three following main components:

1. *Subgradient column generation*. Column generation is a well known technique for solving large-scale linear programs (LPs). The main idea consists of solving the LP over a subset of variables, implicitly considering the other variables and dynamically adding them when optimality conditions are violated.
   We solve the Lagrangean relaxation of PMP by combining subgradient optimization with column generation. Using a special data structure, the subgradient column generation is able to compute (for the first time, at the best of our knowledge) a tight Lagrangean lower bound on instances up to 90,000 vertices.
2. *Core heuristic*. Avella et al. [2] proposed a "core" heuristic consisting of solving first a Lagrangean relaxation, and then a subproblem defined by a subset of the most "promising" variables, selected according to Lagrangean reduced costs. In this paper we propose a revised version of this heuristic which is able to deal with larger instances. Combined with subgradient column generation, the new core heuristic outperforms state-of-the-art approaches on large-scale instances with a relatively large number of medians.
3. *Aggregation heuristic*. On large-scale instances, the core heuristic does not provide good solutions when p is relatively small. To deal with these instances we propose a vertex aggregation procedure.
   Reducing the number of clients by data aggregation is quite a common technique, see for example Hillsman and Rhoda [17], Current and Schilling [6], Hodgson and Neuman [19], Francis et al. [8,9]. However, as noted by Mladenovic et al. [25], the complexity of most PMP methods depends more on the number of facilities than on the number of clients.
   We propose a new procedure based on the aggregation of graph vertices, i.e. we aggregate both facilities and clients. Once aggregation has been performed, the core heuristic becomes effective on the instances with a relatively small number of medians too.

The remainder of the paper is structured as follows. Sections 2–4 describe the main components of the algorithm, namely subgradient column generation, core heuristic and aggregation heuristic, providing their related computational results. Conclusions are discussed in Section 5.

2. Lagrangean column generation

Many successful Lagrangean heuristics for PMP have been presented in the literature (see for instance [3]).

We consider the Lagrangean relaxation obtained by relaxing the assignment constraints (2). Let $\lambda=(\lambda_1,\ldots,\lambda_n)$ be Lagrangean multipliers associated with the relaxed constraints. Then the Lagrangean dual function $\mathcal{L}(\lambda)$ is defined by

$$\mathcal{L}(\lambda)=\min_{(x,y)}\left\{\sum_{ij\in A}d_{ij}x_{ij}-\sum_{j\in I}\lambda_j\left(\sum_{i\in\delta^-(j)}x_{ij}+y_j-1\right):\quad \text{subject to (3)}-(6)\right\}.$$

For any value of multipliers λ Lagrangean dual function provides a lower bound of Z^*, i.e. the lower bound of the optimal value of PMP.

Let $\mu_{ij}(\lambda)=d_{ij}-\lambda_j$ be the Lagrangean reduced cost of the variable x_{ij}, $\mu_{ij}(\lambda)^-=\min\{0,\mu_{ij}(\lambda)\}$ and let

$$\rho_i(\lambda)=\sum_{j\in\delta^+(i)}\mu_{ij}(\lambda)^--\lambda_i$$

be the Lagrangean reduced cost of the variable y_j, where $\delta^+(i)$ is the set of entering arcs of node i. Let them be ordered increasingly, i.e. we are given $i_1,\ldots,i_n$ such that

$$\rho_{i_1}(\lambda)\le\cdots\le\rho_{i_n}(\lambda).$$

The value of the Lagrangean function is obtained by summing-up the best (i.e. smallest) p Lagrangean reduced costs of the variables y and then summing the values of all the multipliers

$$\mathcal{L}(\lambda)=\sum_{k=1}^{p}\rho_{i_k}(\lambda)+\sum_{i\in I}\lambda_i.$$

The Lagrangean function $\mathcal{L}(\lambda)$ can be maximized through a subgradient method, which is based on the iterative formula

$$\lambda^{k+1}=\lambda^k+\gamma_k g(\lambda^k),$$

where $g(\lambda^k)$ is the subgradient at the iteration k computed as

$$g_j(\lambda^k)=1-\sum_{i\in\delta^-(j)}x_{ij}(\lambda^k)-y_j(\lambda^k),$$

where $((x(\lambda^k),y(\lambda^k))$ is a solution of Lagrangean subproblem which corresponds to λ^k. The stepsize γ_k is computed as

$$\gamma_k = \frac{\phi \cdot (1.05 \cdot UB - \mathcal{L}(\lambda^k))}{\|g(\lambda^k)\|_2^2}, \quad (7)$$

where UB denotes the current upper bound, $\|\cdot\|_2$—the Euclidean metric and ϕ—a parameter which is progressively reduced according to some rules, which will be discussed later.

It is known [3] that the best Lagrangean bound is equal to the bound obtained by solving the LP-relaxation, i.e. the linear program obtained by relaxing the integrality constraints.

Let us now analyse the implementation of the subgradient algorithm. Suppose that the distance matrix has been computed and each column is sorted in nondecreasing order, i.e. for each $j \in I$ we have a permutation $\pi(j)$ such that

$$d(\pi_1(j),j) \le d(\pi_2(j),j) \le \cdots \le d(\pi_n(j),j).$$

Let $T(\lambda)$ be a subset of p nodes with the best Lagrangean reduced costs. The Lagrangean function $\mathcal{L}(\lambda)$ for a given λ can be efficiently computed as follows:

1. Initialize $\rho(\lambda) := -\lambda$, $\mathcal{L}(\lambda) := 0$ and $j := 1$;
2. Compute $\mathcal{L}(\lambda) := \mathcal{L}(\lambda) + \lambda_j$ and set $h := 1$;
3. If $d(\pi_h(j),j) \ge \lambda_j$, then go to 6;
4. Compute $\rho_{\pi_h(j)}(\lambda) := \rho_{\pi_h(j)}(\lambda) + d(\pi_h(j),j) - \lambda_j$;
5. If $h < n$ then set $h := h+1$ and go to 3;
6. If $j < n$ then set $j := j+1$ and go to 2;
7. Find $T(\lambda)$ and compute

$$\mathcal{L}(\lambda) := \mathcal{L}(\lambda) + \sum_{i \in T(\lambda)} \rho_i(\lambda).$$

Since computing $\mathcal{L}(\lambda)$ requires keeping into memory the whole distance matrix, hardware limitations should be carefully considered: storing the ordered distance matrix in double precision requires 40 GB of RAM with $n=60{,}000$, a considerable amount of memory for standard workstations. It follows that dealing with larger instances requires to consider more efficient approaches based on a partial storage of the distance matrix.

To this purpose we combine the standard subgradient procedure with delayed column generation. We note that to compute the Lagrangean function and the subgradient, for each $j \in J$ we only need to consider the variables with negative Lagrangean reduced cost, i.e. we need to consider only the first elements of the sorted jth column of the distance matrix, whose values are less than λ_j. For each $j \in I$ we keep into the RAM only the smallest (active) n_0^j elements ($n_0^j < n$). If, at some iteration of the subgradient algorithm, the Lagrangean multiplier λ_j^k exceeds

Table 1
Results of subgradient algorithms.

n	p	Err (%)		Time		Err (%)		Time	
		VNS	SCG	VNS	SCG	VNS	SCG	VNS	SCG
		BIRCH instances of type 1				BIRCH instances of type 3			
10,000	100	0.021	0.011	565	28	0.096	0.109	1738	37
15,000	100	0.213	0.024	2014	59	0.094	0.077	2238	71
20,000	100	0.000	0.007	2497	108	0.181	0.095	2238	138
9600	64	0.023	0.013	969	27	0.123	0.101	939	34
12,800	64	0.015	0.006	1981	49	0.117	0.175	1688	60
16,000	64	0.000	0.009	2233	75	1.890	0.306	2231	105
19,200	64	0.021	0.010	2478	137	0.907	0.728	2483	172
10,000	25	0.065	0.005	989	41	0.834	0.323	889	63
12,500	25	0.049	0.012	1734	60	0.788	0.509	1461	80
15,000	25	0.028	0.011	1932	95	3.099	0.203	2160	139
17,500	25	0.026	0.008	2234	127	1.141	0.944	2231	238
20,000	25	0.001	0.008	2489	198	2.060	0.788	2479	281
		pcb3038				usa13509			
	50	0.172	0.203	198	4	0.042	0.381	2261	92
	60	0.028	0.054	180	4	0.042	0.244	2251	79
	70	0.123	0.141	110	4	0.071	0.146	2335	74
	80	0.056	0.079	158	4	0.028	0.107	2417	77
	90	0.005	0.025	158	3	0.078	0.174	2308	73
	100	0.060	0.086	146	4	0.043	0.209	2301	70
	150	0.039	0.067	92	3	0.072	0.147	2351	60
	200	0.037	0.066	43	3	0.061	0.096	2333	59
	250	0.042	0.056	32	3	0.075	0.102	2401	54
	300	0.036	0.051	30	3	0.393	0.422	2350	55
	350	0.018	0.029	23	3	0.130	0.159	2285	52
	400	0.010	0.021	17	3	0.366	0.387	2266	53
	450	0.015	0.027	18	3	0.667	0.688	2375	51
	500	0.027	0.031	20	3	0.686	0.707	2113	54
	550	0.020	0.024	24	3	0.906	0.927	1320	50
	600	0.054	0.060	28	3	1.087	1.105	1418	50
	650	0.021	0.033	31	3	1.130	1.151	1221	50
	700	0.045	0.058	28	3	0.937	0.960	1391	50
	750	0.061	0.060	33	3	0.817	0.837	1196	50
	800	0.046	0.054	37	3	1.219	1.236	1017	50
	850	0.090	0.073	45	3	1.572	1.591	472	50
	900	0.101	0.067	52	3	1.267	1.285	530	49
	950	0.366	0.116	54	3	1.680	1.699	318	49
	1000	0.134	0.128	66	3	1.282	1.301	271	49

$d(\pi_{n_0^j}(j),j)$, then the jth column is "enlarged" in the RAM, by considering the best subsequent n_1 elements. Then we set $n_0^j := n_0^j + n_1$, and the subgradient algorithm continues.

To prevent the oscillation of Lagrangean multipliers, we adopt the stabilization technique suggested by Hansen et al. [14]. It consists of setting an upper bound to each multiplier, which adjusted dynamically. Initially we set the multipliers to the smallest distance in the corresponding column, i.e. $\lambda_j^0 = d(\pi_1(j),j)$. Further, on each iteration we set upper bounds on the multipliers, i.e. $\lambda_j^k \le ub_j^k$, where

$$ub_j^k = \min_{k \in I}\{d(\pi_j(k),j) : d(\pi_j(k),j) > \lambda_j^{k-1}\},$$

i.e. we set ub_j^k to closest distance above λ_j^{k-1}. Moreover, this technique allows us to keep the number of active elements within reasonable sizes, a multiplier cannot "jump" more than one value in the corresponding column of the distance matrix.

Computational results for subgradient column generation are presented in Table 1.

Computational experiments were carried out on an Intel Core 2Quad CPU 2.6 GHz workstation with 4 GB of RAM, under Windows *XP*64. We did not use multithreads, so computations are limited to a single core. The test bed consists of the same instances considered by Hansen et al. [14]: pcb3038.tsp ($n=3038$) and usa13509.tsp ($n=13{,}509$) from the TSP library[1] and the BIRCH instances kindly provided by Nenad Mladenovic and Dragan Urosevic (n ranges from 10,000 to 20,000).

First we set the initial number of active elements in the columns of the ordered distance matrix n_0^j to take 2 GB of RAM for all active elements. Then it was expanded by setting $n_1=100$. Initially ϕ was set to 2 and it was halved if the lower bound did not improve for 30 iterations. The subgradient stops when ϕ becomes less than 0.005.

We compared our results with those given by the subgradient optimization algorithm of Hansen et al. [14], whose code was kindly provided by the authors.

The notations in the table are the following:

- n is the number of facilities and clients;
- p is the number of medians;
- VNS is the results obtained by the subgradient algorithm of Hansen et al. [14];
- SCG is the results obtained by our subgradient column generation algorithm;
- Err(%) is computed as $((UB-LB)/UB)\cdot 100$, where UB is taken from [14] and LB is obtained by the corresponding algorithms;
- Time is the computation time in seconds.

Computational results show that both algorithms return good quality lower bounds, but subgradient column generation appears to be several times faster.

3. Core heuristic

Once the Lagrangean relaxation has been solved and a lower bound is made available we can compute an upper bound by using a *core selection* approach, which consists of choosing a subset of "promising" variables and then solving a reduced PMP over them. Core selection proved to be effective in solving set covering [5], capacitated facility location [1] and PMP [2] problems. All these papers showed that Lagrangean reduced costs were a good driver to an effective choice of the core.

Let $\overline{\lambda}$ be the "optimal" Lagrangean multipliers returned by subgradient optimization and let $\rho_i(\overline{\lambda})$ and $\mu_{ij}(\overline{\lambda})$ be the optimal Lagrangean reduced costs associated with variables y_i and x_{ij} respectively. The core problem is defined by the subset of variables whose reduced costs are less than a given threshold α for variables y_i, and β for variables x_{ij}. In other words we solve the PMP (1)–(6) where

$$y_i = 0 \quad \forall i \in I(\overline{\lambda},\alpha) = \{i \in I : \rho_i(\overline{\lambda}) > \alpha\}$$

and

$$x_{ij} = 0 \quad \forall ij \in A(\overline{\lambda},\beta) = \{ij : i \in I(\overline{\lambda},\alpha), j \in I\} \cup \{ij : \mu_{ij}(\overline{\lambda}) > \beta\}.$$

The core problem is solved by a commercial MIP solver with a time limit to prevent computation time from being too long. A preliminary computational experience pointed out two main difficulties

- Knowing a good upper bound UB is crucial to get a good convergence of the subgradient procedure (see definition (7) of stepsize γ_k).
- The core problem provides us with a good upper bound on a base of a good solution of subgradient optimization.

To overcome these two obstacles we sequentially repeat the subgradient optimization and solve the core problem in the following way. Let $Z(x,y)$ be the objective function value (1) of the solution (x,y). In the following we summarize the overall procedure:

Core heuristic. *Initialization.* Generate a random feasible solution $(\overline{x},\overline{y})$, set the best upper bound $UB := Z(\overline{x},\overline{y})$ and the best lower bound $LB := -\infty$.

Step 1. Find $\overline{\lambda}$ by the subgradient algorithm, set $LB := \mathcal{L}(\overline{\lambda})$.
Step 2. Construct $I(\overline{\lambda},\alpha)$, $A(\overline{\lambda},\beta)$ and find a solution $(\hat{x},\hat{y})$ of core problem. If $Z(\hat{x},\hat{y}) < UB$ then set $UB := Z(\hat{x},\hat{y})$ and $(\overline{x},\overline{y}) := (\hat{x},\hat{y})$.
Step 3. Continue the subgradient algorithm (i.e. starting from the multipliers from Step 2) finding updated $\overline{\lambda}$ and set $LB := \mathcal{L}(\overline{\lambda})$.
Step 4. Construct $I(\overline{\lambda},\alpha)$, $A(\overline{\lambda},\beta)$ and find a solution $(\hat{x},\hat{y})$ of core problem. If $Z(\hat{x},\hat{y}) < UB$ then set $UB := Z(\hat{x},\hat{y})$ and $(\overline{x},\overline{y}) := (\hat{x},\hat{y})$.
Step 5. Continue subgradient algorithm (i.e. starting from the multipliers from Step 4), find updated $\overline{\lambda}$ and set $LB := \mathcal{L}(\overline{\lambda})$.
Step 6. Return LB, UB and $(\overline{x},\overline{y})$.

A fine tuning of the parameters for each instance in the different steps of the core heuristic can affect the outcomes of the algorithm. In our experiments, we tried to find general settings, able to produce good results for most of the test instances. In Tables 2 and 3 we present the core heuristic results, obtained by using the following parameters:

- In steps 2 and 4, parameters α and β are set to limit the size of the core problem with $|I(\overline{\lambda},\alpha)| = 3p$, $|A(\overline{\lambda},\beta)| = 5n$. To solve the core problem the MIP solver Xpress-Optimizer 18 [7] is used. The running time limit for the core problem is set to 30 s for $p3038$ and 300 s for the other instances.
- In step 1, parameter ϕ is set to 1.5 and divided by 1.01 on each iteration where the lower bound has not been improved.
- In step 3, ϕ is set to 0.1 and updated as said before.
- In step 5, ϕ is set to 0.005 and divided by 1.01 if the lower bound has not been improved after two iterations.

In all steps the subgradient stops when ϕ becomes less 0.001.

[1] www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95

Table 2
Results of core heuristic.

n	p	BUB	Diff (%)			Err (%)		Time		
			GH	VNS	CH	VNS	CH	GH	VNS	CH
		BIRCH instances of type 1								
10,000	100	12,428.5	0.000	0.004	0.000	0.021	0.001	54	786	47
15,000	100	18,639.3	–	0.015	0.000	0.213	0.002	–	3386	101
20,000	100	24,840.3	–	0.000	0.000	0.000	0.001	–	3982	210
9600	64	11,934.8	0.000	0.000	0.000	0.023	0.002	57	1205	56
12,800	64	15,863.8	0.000	0.000	0.000	0.015	0.001	99	2451	84
16,000	64	20,004.6	–	0.000	0.000	0.000	0.001	–	2739	129
19,200	64	24,018.3	–	0.000	0.000	0.021	0.002	–	3698	219
10,000	25	12,455.7	0.000	0.000	0.000	0.065	0.001	95	1091	82
12,500	25	15,597.1	0.000	0.005	8.792	0.049	8.794	151	2073	115
15,000	25	18,949.3	–	0.000	16.677	0.028	16.681	–	2353	175
17,500	25	21,937.4	–	0.000	8.434	0.026	8.437	–	2615	241
20,000	25	25,096.8	–	0.000	10.166	0.001	10.168	–	3055	365
		pcb3038								
	50	507,558.2	0.000	0.144	0.000	0.172	0.034	50	253	28
	60	460,787.5	0.018	0.019	0.000	0.028	0.015	38	257	21
	70	426,093.9	0.000	0.096	0.001	0.123	0.033	47	247	20
	80	397,489.5	0.024	0.015	0.000	0.056	0.046	46	227	23
	90	373,241.9	0.087	0.000	0.000	0.005	0.009	67	239	18
	100	352,618.4	0.037	0.025	0.000	0.060	0.043	38	202	29
	150	281,163.1	0.107	0.012	0.000	0.039	0.035	48	166	37
	200	238,344.2	0.080	0.010	0.000	0.037	0.035	66	277	43
	250	209,214.8	0.031	0.010	0.000	0.042	0.031	71	118	44
	300	187,686.2	0.032	0.012	0.000	0.036	0.027	64	213	16
	350	170,927.0	0.066	0.000	0.004	0.018	0.024	52	206	14
	400	157,027.0	0.032	0.004	0.000	0.010	0.008	72	118	14
	450	145,362.9	0.027	0.004	0.000	0.015	0.013	78	305	11
	500	135,447.4	0.025	0.000	0.000	0.027	0.023	65	254	10
	550	126,825.2	0.064	0.008	0.000	0.020	0.007	80	279	10
	600	119,054.1	0.096	0.043	0.000	0.054	0.006	75	237	10
	650	112,017.7	0.051	0.011	0.000	0.021	0.012	97	221	11
	700	105,822.5	0.032	0.015	0.000	0.045	0.030	85	218	12
	750	100,326.9	0.060	0.022	0.000	0.061	0.027	86	462	14
	800	95,372.5	0.053	0.018	0.000	0.046	0.024	91	494	11
	850	90,981.5	0.077	0.009	0.000	0.090	0.052	60	235	29
	900	86,966.6	0.044	0.001	0.000	0.101	0.054	83	479	28
	950	83,260.3	0.071	0.027	0.000	0.366	0.076	70	469	47
	1000	79,840.1	0.039	0.015	0.000	0.134	0.101	128	531	70

We also compare our results with those returned by the GRASP heuristic of Resende and Werneck [29], whose code is available in the web,[2] but due to the memory limits we could only run this code on instances with $n < 15,000$.

In the tables the following notations are used:

- GR refers to the results obtained by the GRASP heuristic of Resende and Werneck [29];
- VNS is the results obtained by the algorithm of Hansen et al. [14];
- CH refers to the results obtained by our core heuristic;
- *BUB* is the best upper bound obtained among all the algorithms;
- Diff(%) is computed as $((UB-BUB)/UB)\cdot 100$, where *UB* is obtained by the corresponding algorithms.
- Err(%) is computed as $((UB-LB)/UB)\cdot 100$, where *UB* and *LB* are obtained by the corresponding algorithms. The GRASP heuristic does not provide us with the lower bound, therefore these data are omitted.
- Time is the computation time in seconds.

Computational results show that our heuristic is very effective when the number of medians is relatively large. If the number of medians is small, the core problem does not contain any good solution or even it is infeasible. Enlarging the core can slightly improve the results, but considerably increases the computation time.

[2] www.research.att.com/mgcr/popstar/

4. Aggregation heuristic

Computational experience on large scale instances showed that with relatively (to n) small values of p the core problem must be very large to return a good solution, so the core heuristic becomes inefficient. To overcome this problem we propose an *aggregation heuristic* which aggregates sets of vertices into a single one. Other aggregation procedures [6,8,9,17,19] presented in the literature are only based on the aggregation of the clients. The novelty of our approach is that we aggregate sets of vertices of the graph $G(I,A)$ which represent either clients or facilities.

The aggregation heuristic is outlined as follows. The set I is partitioned into $\overline{n}$ nonempty subsets I_k, $k=1,\ldots,\overline{n}$, i.e.

$$I=\bigcup_{k=1}^{\overline{n}} I_k,$$

$$I_k\neq\emptyset,\quad k=1,\ldots,\overline{n},$$

$$I_k\cap I_l=\emptyset,\quad k,l=1,\ldots,\overline{n},\ k\neq l. \tag{8}$$

A representative $i_k\in I_k$ is chosen for each $k=1,\ldots,\overline{n}$. Let $I'=\{i_1,\ldots,i_{\overline{n}}\}$. We aggregate each subset I_k into its representative

Table 3
Results of core heuristic.

n	p	BUB	Diff (%)			Err (%)		Time		
			GH	VNS	CH	VNS	CH	GH	VNS	CH
		BIRCH instances of type 3								
10,000	100	9624.79	0.000	0.050	0.000	0.096	0.002	377	2609	60
15,000	100	15,904.12	–	0.000	21.721	0.094	21.767	–	3495	121
20,000	100	19,989.02	–	0.000	27.923	0.181	27.983	–	3429	222
9600	64	8225.58	0.000	0.055	21.890	0.123	21.912	377	1483	57
12,800	64	10,210.36	0.000	0.062	11.404	0.117	11.412	413	2503	98
16,000	64	13,340.47	–	0.000	23.103	1.890	23.142	–	3169	170
19,200	64	15,207.56	–	0.000	38.697	0.907	38.925	–	3243	229
10,000	25	7203.39	0.000	0.000	11.187	0.834	11.349	316	1016	94
12,500	25	8576.10	0.000	0.339	0.824	0.788	0.956	203	1606	144
15,000	25	9513.64	–	0.000	51.966	3.099	52.041	–	2742	192
17,500	25	12,535.68	–	0.000	37.756	1.141	38.387	–	2803	250
20,000	25	13,052.81	–	0.000	54.339	2.060	54.700	–	3364	364
		usa13509								
	50	157,826,585.11	0.000	0.040	9.699	0.042	9.924	605	2888	140
	60	142,546,915.52	0.000	0.017	7.421	0.042	7.554	775	2587	133
	70	131,089,034.89	0.000	0.071	7.124	0.071	7.142	588	2847	135
	80	122,100,516.25	0.000	0.028	4.618	0.028	4.647	867	2864	151
	90	114,640,547.79	0.000	0.031	1.608	0.078	1.673	849	3012	122
	100	108,009,040.72	0.000	0.025	1.249	0.043	1.313	949	3227	190
	150	86,683,926.02	0.000	0.047	1.896	0.072	1.928	750	2957	124
	200	74,236,026.91	0.000	0.039	1.467	0.061	1.494	1231	3302	168
	250	65,749,066.23	0.009	0.064	0.000	0.075	0.016	1105	3033	194
	300	59,348,169.80	0.009	0.382	0.000	0.393	0.017	850	3330	194
	350	54,548,191.14	0.094	0.115	0.000	0.130	0.021	981	3056	237
	400	50,546,413.01	0.070	0.353	0.000	0.366	0.019	1314	3245	205
	450	47,245,292.59	0.056	0.649	0.000	0.667	0.022	1222	3355	390
	500	44,478,079.98	0.063	0.667	0.000	0.686	0.023	1830	3105	250
	550	42,047,061.02	0.110	0.900	0.000	0.906	0.009	925	2312	192
	600	39,959,144.28	0.096	1.066	0.000	1.087	0.025	1085	2410	420
	650	38,122,254.72	0.108	1.111	0.000	1.130	0.024	850	2213	187
	700	36,477,346.99	0.115	0.917	0.000	0.937	0.025	1004	2389	419
	750	34,996,447.36	0.072	0.795	0.000	0.817	0.026	1399	2193	435
	800	33,642,324.43	0.110	1.205	0.000	1.219	0.017	1290	2015	203
	850	32,409,918.19	0.084	1.557	0.000	1.572	0.018	1191	1468	303
	900	31,281,915.65	0.077	1.246	0.000	1.267	0.024	1605	1526	309
	950	30,240,777.71	0.096	1.656	0.000	1.680	0.027	1046	1302	215
	1000	29,276,485.23	0.132	1.255	0.000	1.282	0.030	1225	1267	190

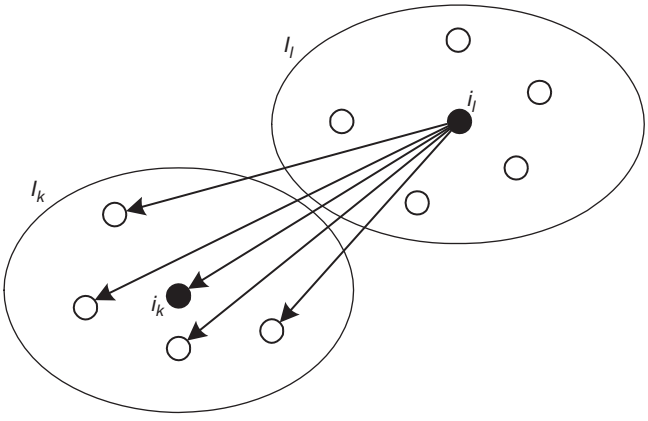

Fig. 2. Arcs which are used in computing $d'(i_l,i_k)$.

i_k and define the distances between the nodes in I' according to the following rule:

$$d'(i_l,i_k)=\sum_{j\in I_k} d(i_l,j),$$

i.e. we assume that the distance between i_k and i_l is the sum of the distances from i_l to all the elements in I_k. It is illustrated in Fig. 2. We note that, in general, $d'(i_k,i_k)\neq 0$ and $d'(i_l,i_k)\neq d'(i_k,i_l)$.

The MIP formulation of reduced PMP over the set I' with distances $d'(\cdot)$ is

$$\min_{(x',y')} \sum_{l=1}^{\overline{n}}\sum_{k=1,k\neq l}^{\overline{n}} d'(i_l,i_k)x'_{lk}+\sum_{l=1}^{\overline{n}} d'(i_l,i_l)y'_l, \tag{9}$$

$$\sum_{l=1,l\neq k}^{\overline{n}} x'_{lk}+y'_k=1 \quad \forall k=1,\ldots,\overline{n}, \tag{10}$$

$$x'_{lk}\le y'_l \quad \forall l,k=1,\ldots,\overline{n}, \tag{11}$$

$$\sum_{l=1}^{\overline{n}} y_l=p, \tag{12}$$

$$y'_l\in\{0,1\} \quad \forall l=1,\ldots,\overline{n}, \tag{13}$$

$$x'_{lk}\in[0,1] \quad \forall l,k=1,\ldots,\overline{n}. \tag{14}$$

Let $(\overline{x}',\overline{y}')$ be a solution of problems (9)–(14). It can be transformed into a solution $(\overline{x},\overline{y})$ of original problems (1)–(6) as follows:

If $\overline{y}'_l=1$ then $\overline{y}_{i_l}=1$ and $\overline{x}_{i_lj}=1$ for all $j\in I_l, j\neq i_l$.
If $\overline{x}'_{lk}=1$ then $\overline{x}_{i_lj}=1$ for all $j\in I_k$.

In other words, if i_k is assigned to the median i_l, then all elements of I_k and I_l are also assigned to the median i_l.

A crucial issue has to be addressed to make the procedure effective: how to partition I? We suggest to consider the partition problem as a PMP with a relatively large number of medians. A solution of problems (9)–(14) can be found by the core heuristic described in Section 3, which is effective when p is large. It is evident that enlarging the size of the reduced problem leads to a better solution of the original problem, but on the other hand the size must be limited in order to rapidly find a good solution of the reduced problem. We can observe by the computational experiments that PMP with up to $|I| = 5000$ can be easily tackled by our core heuristic, so we set $\overline{n} = 5000$.

Summarizing, the complete aggregation heuristic consists of the following steps:

1. Find a solution of PMP with 5000 medians by the core heuristic and construct a partition using this solution.
2. Find the solution of the reduced problems (9)–(14) by the core heuristic and recover the solution of the original problems (1)–(6).
3. Compute the lower bound of original problems (1)–(6) by the subgradient algorithm.

For the instances where Err is larger than 1%, we run the aggregation procedure. The size of the core problem for solving the reduced PMP is chosen according to the rule $|I(\overline{\lambda},\alpha)| = 2p$, $A(\overline{\lambda},\beta) = 3n$. Then a lower bound is obtained by running a subgradient algorithm with the settings described in Section 2. The results are given in Table 4. The results of the aggregation heuristic are marked with AH. As we can see it is very effective: there is only one instance where the difference between the upper and lower bounds remained slightly greater than 1%.

Table 5 presents the results on large BIRCH instances. The algorithm of Hansen et al. [14] has been able to obtain only upper bound for instances of type 1, while our approach has found upper and lower bounds for all of them, keeping the solution quality better than 1%.

5. Conclusions

We introduce a new Lagrangean heuristic which is able to deal with very large-scale p-median problems, as those arising in cluster analysis. Clustering instances of the p-median problem usually show small duality gaps, and subgradient optimization of

Table 5
Results on large BIRCH instances.

n	p	UB		Err (%)	Time	
		VNS	AH	AH	VNS	AH
		BIRCH instances of type 1				
25,000	25	31,363.6	31,282.6	0.181	206	447
36,000	36	45,115.6	45,226.3	0.261	590	780
49,000	49	61,384.1	61,569.7	0.319	818	1216
64,000	64	79,987.3	80,337.4	0.369	1527	2258
30,000	25	37,564.1	37,617.1	0.161	321	559
43,200	36	54,191.4	54,305.8	0.226	767	1003
58,800	49	73,626.8	73,854.7	0.324	1454	1691
76,800	64	95,989.1	96,393.4	0.384	2931	2834
35,000	25	43,902.1	43,972.1	0.180	569	768
50,400	36	63,169.2	63,329.2	0.266	1185	1472
68,600	49	85,833.6	86,082.0	0.300	1787	2441
89,600	64	112,059.2	112,485.2	0.393	3678	4501
		BIRCH instances of type 3				
25,000	25		17,718.6	0.210		527
36,000	36		27,476.1	0.682		913
49,000	49		44,282.5	0.663		1760
64,000	64		58,991.5	0.467		2624
30,000	25		21,865.1	0.610		832
43,200	36		32,391.6	0.575		1873
58,800	49		50,985.1	0.394		2692
76,800	64		66,944.7	0.816		4393
35,000	25		24,833.7	0.183		972
50,400	36		38,162.3	0.407		2297
68,600	49		62,007.4	0.981		3556
89,600	64		79,245.3	0.978		5779

Table 4
Results of aggregation heuristic.

n	p	BUB	Diff (%)			Err (%)		Time		
			GH	VNS	AH	VNS	AH	GH	VNS	AH
		BIRCH instances of type 1								
12,500	25	15,597.12	0.000	0.005	0.086	0.049	0.096	151	2073	162
15,000	25	18,949.26	–	0.000	0.117	0.028	0.125	–	2353	194
17,500	25	21,937.40	–	0.000	0.103	0.026	0.111	–	2615	228
20,000	25	25,096.82	–	0.000	0.130	0.001	0.136	–	3055	312
		BIRCH instances of type 3								
15,000	100	15,904.12	–	0.000	0.312	0.094	0.383	–	3495	186
20,000	100	19,989.02	–	0.000	0.359	0.181	0.462	–	3429	289
9600	64	8225.58	0.000	0.055	0.163	0.123	0.190	377	1483	123
12,800	64	10,210.36	0.000	0.062	0.228	0.117	0.331	413	2503	171
16,000	64	13,340.47	–	0.000	0.172	1.890	0.375	–	3169	240
19,200	64	15,207.56	–	0.000	0.272	0.907	0.999	–	3243	321
10,000	25	7203.39	0.000	0.000	0.058	0.834	0.330	316	1016	160
12,500	25	8576.10	0.000	0.339	0.097	0.788	0.286	203	1606	188
15,000	25	9513.64	–	0.000	0.076	3.099	0.256	–	2742	261
17,500	25	12,535.68	–	0.000	1.044	1.141	2.105	–	2803	359
20,000	25	13,036.63	–	0.124	0.000	2.060	0.674	–	3364	480
		usa13509								
	50	157,826,585.11	0.000	0.040	0.112	0.042	0.506	605	2888	207
	60	142,546,915.52	0.000	0.017	0.127	0.042	0.421	775	2587	204
	70	131,089,034.89	0.000	0.071	0.149	0.071	0.221	588	2847	193
	80	122,100,516.25	0.000	0.028	0.176	0.028	0.289	867	2864	203
	90	114,640,547.79	0.000	0.031	0.182	0.078	0.323	849	3012	231
	100	108,009,040.72	0.000	0.025	0.263	0.043	0.412	949	3227	217
	150	86,683,926.02	0.000	0.047	0.346	0.072	0.445	750	2957	201
	200	74,236,026.91	0.000	0.039	0.438	0.061	0.487	1231	3302	204

Lagrangean relaxation can return good lower bounds. When solving large-scale instances, the bottlenecks are data I/O requirements. To overcome this problem we present a subgradient column generation approach and an aggregation procedure which allows us to compute an upper bound on a reduced problem. Embedding these procedures into a core heuristic gives excellent results which are highly competitive with the state-of-the-art approaches on instances with up to 20,000 nodes.

The most effective existing heuristics for large-scale p-median problems are the GRASP of Resende and Werneck [29] and the PDVNDS of Hansen et al. [14]. Our computational experiments point out that:

1. Subgradient column generation provides lower bounds with the same quality as the standard subgradient algorithm presented in PDVNDS, but it is much faster.
2. The core heuristic without the aggregation procedure is faster than GRASP and PDVNDS and can get a better solution on instances with relatively large p.
3. On instances with small p, the aggregation heuristic is able to obtain high-quality solutions faster than GRASP and PDVNDS in most cases.

Moreover, we show that our approach can deal with instances up to 89,600 nodes on a modern workstation obtaining high-quality solution in reasonable time. At the best of our knowledge, these are the largest instances tested in the literature.

All the instances and the code are publicly available and can be downloaded from iv.icc.ru/Papers.html.

References

[1] Avella P, Boccia M, Sforza A, Vasilyev I. An effective heuristic for large-scale capacitated facility location problems. Journal of Heuristics 2009;15(6):597–615.

[2] Avella P, Sassano A, Vasil'ev I. Computational study of large-scale p-median problems. Mathematical Programming 2007;109(1):89–114.

[3] Beasley JE. Lagrangean heuristics for location problems. European Journal of Operational Research 1993;65(3):383–99.

[4] Beltrán C, Tadonki C, Vial JPh. Solving the p-median problem with a semi-Lagrangian relaxation. Computational Optimization and Applications 2006;35(2):239–60.

[5] Caprara A, Fischetti M, Toth P. A heuristic method for the set covering problem. Operations Research 1999;47(5):730–43.

[6] Current JR, Schilling DA. Elimination of source A and B errors in p-median location problems. Geographical Analysis 1987;19(2):95–110.

[7] Dash Optimization. Xpress-optimizer reference manual, Release 18; 2007.

[8] Francis RL, Lowe TJ, Tamir A. Aggregation error bounds for a class of location models. Operations Research 2000;48(2):294–307.

[9] Francis RL, Lowe TJ, Tamir A. Worst-case incremental analysis for a class of p-facility location problems. Networks 2002;39(3):139–43.

[10] Garcia S, Labbé M, Marin A. Solving large p-median problems with a radius formulation. INFORMS Journal on Computing, published online, doi:10.1287/ijoc.1100.0418.2010.

[11] Hakimi SL. Optimum distribution of switching centers in a communication network and some related graph theoretic problems. Operations Research 1965;13(3):462–75.

[12] Hakimi SL. Optimum location of switching centers and the absolute centers and medians of a graph. Operations Research 1964;12(3):450–9.

[13] Hansen P, Brimberg J, Urošević D, Mladenović N. Primal–dual variable neighborhood search for the simple plant-location problem. INFORMS Journal on Computing 2007;19(4):552–64.

[14] Hansen P, Brimberg J, Urošević D, Mladenović N. Solving large p-median clustering problems by primal–dual variable neighborhood search. Data Mining and Knowledge Discovery 2009;19(3):351–75.

[15] Hansen P, Jaumard B. Cluster analysis and mathematical programming. Mathematical Programming 1997;79(1–3):191–215.

[16] Hansen P, Mladenović N, Pérez-Britos D. Variable neighbourhood decomposition search. Journal of Heuristics 2001;7(4):335–50.

[17] Hillsman EL, Rhoda R. Errors in measuring distances from populations to service centers. The Annals of Regional Science 1978;12(3):74–88.

[18] Hinojosa Y, Kalcsics J, Nickel S, Puerto J, Velten S. Dynamic supply chain design with inventory. Computers and Operations Research 2008;35(2):373–91.

[19] Hodgson MJ, Neuman S. A GIS approach to eliminating source C aggregation error in p-median models. Location Science 1993;1(2):155–70.

[20] Jain AK, Murty MN, Flynn PJ. Data clustering: a review. ACM Computing Surveys 1999;31(3):264–323.

[21] Kariv O, Hakimi L. An algorithmic approach to network location problems. II. The p-medians. SIAM Journal on Applied Mathematics 1979;37(3):539–60.

[22] Klastorin TD. The p-median problem for cluster analysis: a comparative test using the mixture model approach. Management Science 1985;31(1):84–95.

[23] Kochetov Yu, Ivanenko D. Computationally difficult instances for the uncapacitated facility location problem. In: Ibaraki T, editor. Metaheuristics: progress as real solvers. Springer; 2005. p. 351–67.

[24] Mirchandani P, Francis R, editors. Discrete location theory. New York: Wiley-Interscience; 1990.

[25] Mladenović N, Brimberg J, Hansen P, Moreno-Pérez JA. The p-median problem: a survey of metaheuristic approaches. European Journal of Operational Research 2007;179(3):927–39.

[26] Mulvey JM, Crowder HP. Cluster analysis: an application of Lagrangian relaxation. Management Science 1979;25(4):329–40.

[27] Pacheco JA, Álvarez A, Casado S, Alegre JF. Heuristic solutions for locating health resources. IEEE Intelligent Systems 2008;23(1):57–63.

[28] Rao MR. Cluster analysis and mathematical programming. Journal of the American Statistical Association 1971;66(335):622–6.

[29] Resende MGC, Werneck RF. A hybrid heuristic for the p-median problem. Journal of Heuristics 2004;10(1):59–88.

[30] Senne ELF, Lorena LAN, Pereira MA. A branch-and-price approach to p-median location problems. Computers and Operations Research 2005;32(6):1655–64.

[31] Senne ELF, Lorena LAN. Lagrangean/surrogate heuristics for p-median problems. In: Laguna M, Gonzalez-Velarde JL, editors. In computing tools for modeling, optimization and simulation: interfaces in computer science and operations research. Kluwer Academic Publishers; 2001. p. 115–30.

[32] Taillard ED. Heuristics methods for large centroid clustering problems. Journal of Heuristics 2003;9(1):51–73.

[33] Vinod HD. Integer programming and the theory of grouping. Journal of the American Statistical Association 1969;64(326):506–19.