A Computational Comparison of Different Algorithms for Very Large p-median Problems

Pascal Rebreyend[1](✉), Laurent Lemarchand[2], and Reinhardt Euler[2]

[1] School of Technology and Business Studies, Dalarna University, Falun, Sweden
prb@du.se

[2] Lab-STICC/UBO, Université Européenne de Bretagne, Brest, France
{laurent.lemarchand,reinhardt.euler}@univ-brest.fr

Abstract. In this paper, we propose a new method for solving large scale p-median problem instances based on real data. We compare different approaches in terms of runtime, memory footprint and quality of solutions obtained. In order to test the different methods on real data, we introduce a new benchmark for the p-median problem based on real Swedish data. Because of the size of the problem addressed, up to 1938 candidate nodes, a number of algorithms, both exact and heuristic, are considered. We also propose an improved hybrid version of a genetic algorithm called `impGA`. Experiments show that `impGA` behaves as well as other methods for the standard set of medium-size problems taken from Beasley's benchmark, but produces comparatively good results in terms of quality, runtime and memory footprint on our specific benchmark based on real Swedish data.

1 Introduction

Facility location problems consider a set of demand points to be served from a set of possible locations. Solution quality takes into account costs for associating demand points to locations and also costs of choosing a particular location. In the p-median version, the latter is not considered, i.e., we can describe the p-median problem as finding a set of p facilities such that the sum of distances between demand points and the closest facility is minimized. The p-median problem has been introduced by Hakimi [15] who describes its basic properties.

Previous approaches to the p-median problem [1,11,14] have included numerical tests using Beasley's benchmark [7]. The largest graph of this benchmark has 900 candidate nodes on which we can locate facilities. Graphs are generated with a uniform distribution of the demand, which is not representing accurately a true problem since the population in most countries and therefore the demand is not uniformly distributed due to the presence of natural factors like islands, lakes, mountains, rivers,..., and the concentration areas related to urbanization. Further, to represent most regional or national location problems with just 900 candidate nodes is rather limited and implies a high degree of problem simplification together with a loss of information accuracy. Another set of abstract graphs has been used by Avella [4], with the same drawback of uniform distribution of the demand.

G. Ochoa and F. Chicano (Eds.): EvoCOP 2015, LNCS 9026, pp. 13–24, 2015.
DOI: 10.1007/978-3-319-16468-7_2

Due to these two limitations of the Beasley benchmark, one of our interest in this paper is to introduce and use a new set of problem instances based on real data. This new set is based on Swedish real data. Sweden is a good candidate to test methods for the p-median problem since the distribution of population is not uniform as we can see in Fig. 1. The size of the country ($449,964\,\text{km}^2$) is big enough to test algorithms designed for country-related problems. Sweden exhibits also some particular characteristics which make a clear difference between an abstract graph and those obtained from real data such as the presence of natural barriers i.e., lakes or mountains. The non-uniform distribution of the population is also taken into account by distinguishing between demand nodes and candidate nodes. Demand nodes indicate where people are living while candidate nodes are possible locations for a facility. The number of facilities p will vary from 10 to 100 in our cases to cover different practical p-median problems such as locating universities ($p = 10$), courts, public hospitals ($p = 100$),.... Locating hospitals is the practical problem we focus on, without limitation on the capacity of a facility. This corresponds to most of the computational experiments done so far [22].

Some tests on larger graphs have been carried out by Avella [4] using the Birch set of abstract graphs (with up to 89 000 nodes). Birch graphs have similar to those of Beasley a uniform distribution of the demand but points are grouped into clusters. Real graphs with up to 67,000 nodes have been used by Rebreyend et al. [21] to investigate effects of the quality of different road networks on the p-median problem. But in their case only one heuristic method was tested and the number of candidate nodes to locate on was high in comparison to the geographical area as was the number of nodes representing locations of population, since data represent only a single Swedish province. The authors of [21] conclude that fewer nodes lead to better results also because they are using an approximate method.

Sweden has an asymmetrical distribution of the population and natural barriers are spread all over the country. Therefore, we need to use the road distances since Euclidian distances may lead to poor results [9].

The p-median problem is described in the next section. Section 3 presents previous work and algorithms related to the p-median problem. A detailed explanation on the algorithms we have used is given in Sect. 4. Section 5 describes our new algorithm. Section 6 presents the data used. Results in terms of quality, runtime and memory footprint for the tested algorithms in Sects. 7, and Sect. 8 concludes the paper.

2 The *p*-median Problem

In the rest of this paper, the following terms are used:

- N for the number of candidate nodes (number of possible locations for a facility),
- D for the number of demand nodes,
- p for the number of facilities to allocate.

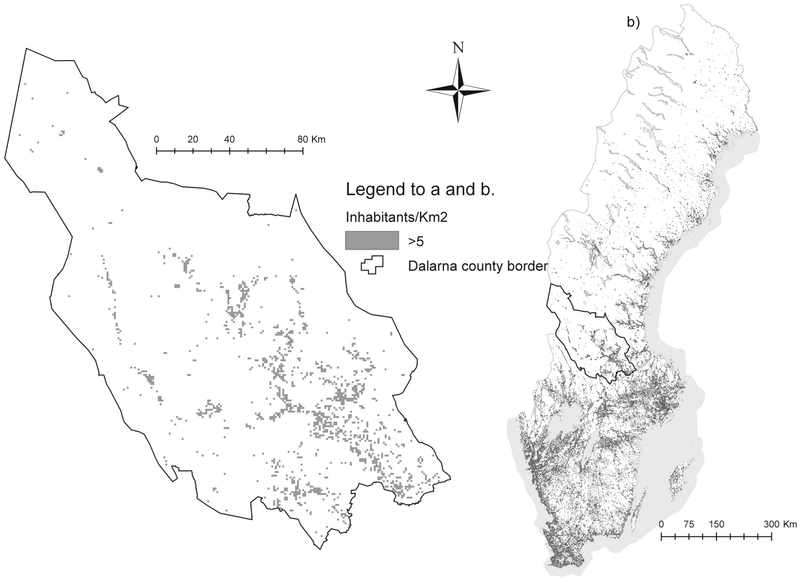

Fig. 1. Distribution of two populations, Dalecarlia province on the left, Sweden on the right

A common formulation of the p-median problem due to Revelle and Swain [23] is the following:

$$\text{minimize} \quad \sum_{i=1}^{D}\sum_{j=1}^{N} h_i d_{ij} Y_{ij} \tag{1}$$

$$\text{subject to} \quad \sum_{j=1}^{N} Y_{ij} = 1 \qquad \forall i, 1 \leq i \leq D \tag{2}$$

$$\sum_{j=1}^{N} X_j = p \tag{3}$$

$$Y_{ij} - X_j \leq 0 \qquad \forall i, j,\ 1 \leq i \leq D,\ 1 \leq j \leq N \tag{4}$$

$$X_j \in \{0, 1\} \qquad \forall j,\ 1 \leq j \leq N \tag{5}$$

$$Y_{ij} \in \{0, 1\} \qquad \forall i, j,\ 1 \leq i \leq D,\ 1 \leq j \leq N \tag{6}$$

where

- h_i is the weight of the demand of customer i,
- d_{ij} is the distance between customer i and facility j,
- X_i is a decision variable, indicating whether facility i is selected or not,
- Y_{ij} is a decision variable, indicating whether customer i is served by facility j or not,
- p is the number of facilities to be selected.

Equation (2) ensures that a customer is served by exactly one facility. The number of facilities is fixed by Eq. (3). Constraint set (4) reflects the objective that demand points are assigned only to a selected facility. Binary decision variables X and Y are defined by Eqs. (5) and (6).

At least one optimal location exists if the facilities are located on the nodes of the graph only [15]. Kariv and Hakimi [16] have shown that the p-median problem is NP-Hard.

3 Related Work

Reese [22] has recently published a bibliography on the main methods used for solving the p-median problem. A survey of metaheuristics has been published by Mladenovic [20]. The earliest solution techniques mentioned are enumeration-based or heuristics such as vertex-substitution [3,18]. Simulated Annealing (SA) based approaches have also been applied to the p-median problem [1,9,20]. Genetic algorithms (GA) have been used in [2,11]. Some approaches rely on hardware to improve runtimes, and thus enlarge the applicability of algorithms. Parallel versions of GA-based approaches have also been implemented [8]. More specifically, GPU-based implementations of the Vertex substitution heuristic [17] or of the Volume algorithm onto multi-core systems [14] can lead to impressive speedups. Exact methods like 0-1 programming have also been proposed [5].

In this paper we focus on sequential algorithms that are suitable for very large problem instances. A detailed description will be given in the next section.

4 Tested Algorithms

4.1 CPlex

The p-median problem can be formulated as a 0-1 programming problem (BP) and then be solved by a Mixed Integer Problem (MIP) solver, using a branch and cut approach. In our tests, we have used the CPlex software from IBM (version 12.6, Linux 64 bits) to test the BP approach. Some parameters of the solver have been tuned in order to adapt CPlex to work on large problem instances, i.e., removing default computation time limits, allowing intermediate data storage, and tuning branch & cut search tree strategies according to [13]. In the following we will refer to this implementation as CPLEX.

4.2 Volume

Barahona and Anbil have given in [6] a description of the Volume algorithm. This algorithm solves Mixed Integer Programs (MIP) by working on the dual of a linear problem using the sub-gradient method [14]. At each iteration approximations of the primal variable values are computed in addition. By working both on the primal and dual problem, the Volume algorithm computes lower and upper bounds for a given problem instance very efficiently, thanks to the subgradient method. The Volume algorithm has been successfully tested on the p-median problem by specializing it to the associated LP-relaxation. The relaxed solution is then exploited by fast heuristics to compute an integral solution of the original p-median problem [10,14].

The version of the Volume algorithm we used[1] to solve the p-median problem is based on the formulation given in Sect. 2, except that constraint (4) is replaced by the following [10]:

$$\sum_{i \neq j} Y_{ij} + X_j = 1, \forall j, 1 \leq j \leq N \tag{7}$$

$$Y_{ij} \leq X_j, \forall i, j, 1 \leq i \leq D, 1 \leq j \leq N \tag{8}$$

Equation (7) indicates that either a node is selected, or it is connected to one candidate. Equation 8 is identical to constraint (4). As a first step for solving a p-median problem instance, the LP-relaxation of the dual problem is formulated, and the heuristic of Bourges-Cleraux [10] is used to find a feasible integer solution from the vector of real numbers found: the p highest values of this vector are selected as the set of locations used. The second step is to select for each customer the closest facility. This is done by sorting edges of the graph according to their distance, from the shortest to the highest and going through them. The next step is to go through all edges in this order. If an edge connects a selected location to a customer with no location assigned to, the corresponding location is assigned to this customer. Altogether, the complexity of this heuristic is $O(m \log m)$ if m is the number of edges since a sort on all edges is done. Default values have been used for the different parameters.

4.3 Simulated Annealing

Al-Khedhairi presents a general version of a Simulated Annealing (SA) algorithm for the p-median problem [1]. Carling et al. have later used a similar approach to solve the p-median problem in the real context of a Swedish province [9,21].

In this paper, we use Carling et al.'s algorithm. The starting point is a random solution of the p-median problem instance. The neighbourhood of a solution s is defined as the set of all solutions s' in which one of the selected nodes of s has been replaced by another candidate node. All the candidate nodes have the same probability to be chosen.

The initial temperature of the SA is fixed to 400°. At every iteration the temperature is multiplied by 0.95. A main concern with simulated annealing is the risk to get stuck at a frozen state. To detect such a situation and reheat the system, we proceed as follows: if 10 consecutive iterations do not result in any improvement, the following formula is applied to modify the temperature t: $t = t * 3^{\beta}$. The initial value of β has been set to 0.5 after some experiments. If between two modifications no solution has been accepted, β is increased by 0.5.

[1] We thank C. Cleraux for providing us the code.

As soon as a solution is accepted, β is reset to its value of 0.5. These parameters have been set up according to previous experiments done with the p-median problem [19,21].

4.4 Genetic Algorithm

Several researchers have proposed genetic algorithms (GA) for solving the p-median problem [2,11]. Most of them use a classical string representation, i.e. each chromosome is represented as a single string of length p embedding the index of the selected facilities or nodes. In our experiments, we are using a genetic algorithm based on Correa et al. [11], that has proven its efficiency for large scale instances. We add the constraint that in all chromosomes no facility is duplicated. The initial population of our algorithm is randomly generated and all the candidate locations have the same probability to be chosen.The crossover used is the one described in [11]. It takes as input 2 chromosomes (called A and B) and generates two new offsprings (A' and B') which replace the ones used as input in the global population. The two new chromosomes are generated by the following procedure:

1. Numbers that appear in both A and B are copied into A' and B'
2. Two exchange vectors EA and EB are computed as follows: EA (resp. EB) are integers in A (resp. B) which are not a member of B (resp. A); (obviously EA and EB have the same size).
3. Let r be a random integer between 0 and the size of EA.
4. r integers are randomly selected from EA and EB and exchanged against each other.
5. EA (resp. EB) is copied into A' (resp. B')

We also reuse the mutation of Correa et al. [11]. For a given chromosome, let r be a random integer between 1 and $p/10$. Pick at random r integers from A and for each of them choose at random an integer not in A to replace it. The resulting genetic algorithm will be called `basic-GA` in the rest of the paper.

Correa et al. [11] also introduce a local search method in their GA by means of a new mutation called *hypermutation*. This operation works as follows for a given chromosome: the heuristic loops on all selected facilities represented by the chromosome. Each selected facility is successively replaced by a facility not already present in the solution, and for every new chromosome produced in this way, the best solution with respect to fitness is kept. `hyper-GA` is the version of the genetic algorithm which uses hypermutation.

5 Improved Genetic Algorithm

In this section an improved genetic algorithm called `imp-GA` is proposed. It is designed to perform well on very large p-median problems to be described in Sect. 6. This improved algorithm is based on the `hyper-GA` of Correa et al. [11]. In `hyper-GA`, the local search is done via hypermutation and has a complexity of $O(pN)$. Instead, `imp-GA` has a lower complexity for its local search which is shown below as Algorithm 1. It selects a chromosome and returns the best solution found in its neighborhood.

We have designed a new version of the hypermutation operation which turns out to work well on very large problem instances. Its main idea is to diminish the space of the local search in order to reduce the computing cost of hypermutation as soon as we increase the graph size. The new hypermutation has a complexity of $O(N)$ since only a small, fixed number F of selected candidates are considered for replacement.

```
begin
    Let $A$ be the chromosome;
    Choose randomly $F$ facilities that appear in $A$;
    foreach facility among the $P$ chosen nodes do
        for $0 \leq i \leq N$ do
            replace the selected facility by facility $i$ (if $i$ is not already in $A$);
            if fitness($A'$) < fitness($A$) then
                $A \longleftarrow A'$ ;
            end
        end
    end
end
```

Algorithm 1. The new hypermutation operation

In our experiments, we have limited the number of iterations to 100, and the parameter F has been set to 5. Since the new hypermutation has a lower complexity, it is run at each iteration of the genetic algorithm in contrast to that of the hypermutation of the previously described genetic algorithm whose probability of being used for a given generation is only 0.5 %. The new hypermutation has only a reduced local search space, but this leads to small computation times and therefore we can apply it more often. All other parameters have been set to the same values as those used in [11].

We have applied two selections. Correa is using *rank* selection. The *biased roulette wheel* is another selection scheme commonly used [12]. These two selection methods have been tested and experiments have shown that the roulette wheel gives better results than the rank selection. Therefore, for the rest of the article we will only consider the biased roulette wheel selection. It works as follows: we have n chromosomes. Each of them has a fitness value f_i,$i = 1, ..., n$. The probability for an individual i to be chosen is $p(i) = \frac{Max - f_i}{\sum_i Max - f_i}$, where Max is the highest fitness value among all chromosomes of the population. Once the probability of each individual is computed, the selection will choose randomly n new candidates according to their probabilities.

Table 1. Results for the Beasley benchmark (40 graphs)

	CPlex	SA	Genetic algorithms			Volume
			basic-GA	hyper-GA	imp-GA	
			1000 iterations	1000 iterations	100 iterations	
# optimal	40	11	20	20	23	5
Deviation in %	0.0	2.35	0.1	0.14	0.2	4.2
Total time (secs)	64329	2294	70	5246	4862	112

6 Data

For our experiments, we have used two different instance sets. One is the set of graphs from Beasley [7] which is commonly used for testing p-median algorithms. The number of candidate nodes varies from 100 to 900. Before running the tested algorithms, we have precomputed for each instance the matrix of distances.

The second instance set is based on official Swedish data. The distance matrix is derived from the National Road Database (NVDB) of the Swedish Road Administration. From the database which stores all road segments, a graph representing the road network is built by identifying crossings using the x,z,y coordinates [19]. Islands or strongly connected components are detected and virtual links are added to simulate ferries. Then, the graph is cleaned from data unnecessary for the p-median problem such as dead-end roads with no people living along, or points which are neither crossings nor demand nodes. After this process, we still have several millions of nodes. According to Hakimi [16], these points are the set of points we should consider for the p-median location nodes. This graph will be used to compute distances between location and demand nodes. The set of demand nodes is provided by Statistic Sweden [19] and consists of $188,325$ weighted points. Each point indicates how many persons at the age of 20 to 64 are living in a square centered around this point (in 2012). The size of the square varies from 250 by 250 m to 1 km by 1 km depending on its location. The total size of the population is $5,411,373$ persons. The average number of people represented by a point is thus 29 and the highest one $2,302$.

Observe that the approach used by Carling et al., and Rebreyend et al. [9,21] for the Swedish province of Dalecarlia cannot be used directly due to the lack of information on the road type. In order to reduce the number of candidate nodes, and to have a good tradeoff between quality and accuracy, we have therefore chosen as candidate node the centers of 1938 Swedish settlements. Some further arguments can be put forward to justify this approach. First, according to previous results of [9,21], since p will be small (less than 100 for all Sweden), a smaller number of candidate points may not degrade the solution found. Another argument is that all hospitals in Sweden are located less than 3 Km to the city center. Finally, settlements represent well the area where most of the people are located and therefore highly densely populated areas will have more candidate points than sparse areas which is important when the distribution of the demand is non-uniform. Smaller tests were created by restricting locations to the Dalarna province (named Dalarna) and to both the Dalarna and Gävleborg

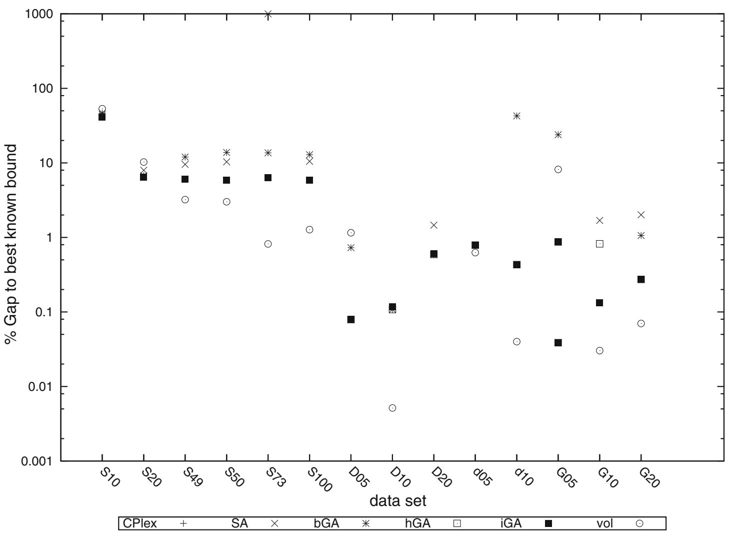

Fig. 2. Compared quality of the solutions obtained by different algorithms on the Swedish benchmark. Quality of a solution is expressed as the percentage of the gap with respect to the best known bound

provinces (named DalGavle). A third smaller case was created by taking only 54 candidate nodes for Dalarna instead of the 108 settlements.

7 Results

We aim to compare the different algorithms described in Sect. 4, with the one we designed, imp-GA, applied to the benchmark set of Beasley [7], and then to the real Swedish data set described in Sect. 6. We want to check the quality of solution and of the runtimes, and also to verify if similar results are obtained for standard benchmark and real-life testcases. For the algorithms' settings, we have used values by default from the corresponding software for CPlex and the Volume algorithm. Since SA is sensitive to its starting point, results are the best of 4 runs from different starting points, each run having 2, 000, 000 iterations.

Table 1 synthesizes our results for the Beasley benchmark set. Average results in terms of quality are perfect for CPlex, solving all of the cases to optimality: the benchmark scale (900 nodes maximum) is affordable by up to date software and hardware. Concerning heuristic approaches, imp-GA solves the most important set of cases to optimality (23) and is always close to, with a standard deviation of 0.2 %. Other GAs also perform well, with similar standard deviations. SA is worst, with a deviation of 2.35 %. The Volume algorithm is the less accurate approach for those cases, with a standard deviation of 4.2 %, but it is the fastest in term of global runtime: 112 s vs 4862 s for imp-GA, and 64329 secs for CPlex. Globally, these results show that all algorithms are applicable to "small" cases solvable to optimality with an exact approach.

Table 2. Results for the Swedish benchmark. (*) excluding Sweden subset

Problem	N	p	Best (*known lower bound*)	CPLEX	SA	Genetic algorithms			Volume
						basic-GA 1000 iterations	hyper-GA 1000 iterations	imp-GA 100 iterations	
Sweden	1938	10	40553	MEM	58590	59398	TIME	57332	62163
		20	35338	MEM	38154	37720	TIME	37614	38965
		49	19848	MEM	21744	22228	TIME	21042	20487
		50	19633	MEM	21661	22346	TIME	20788	20225
		73	15633	MEM	171961	17766	TIME	16621	15761
		100	12930	MEM	14291	14598	TIME	13689	13095
Dalarna	108	5	19863	19879	19879	20008	19879	19879	20093
		10	11660	11673	11673	11674	11673	11674	11661
		20	7237	7280	7343	7280	7280	7281	7237
Dalarna54	54	5	20910	21075	21075	21075	21075	21075	21041
		10	12270	12323	12323	17510	12323	12323	12275
		20	8398	8472	8472	10404	8472	8472	8398
DalGavle	195	5	27937	27948	27948	27948	27948	27948	30221
		10	17486	17510	17782	17510	17630	17510	17492
		20	10294	10323	10502	10404	10323	10323	10302
% Quality std dev.				0.3(*)	6.7	11.8	0.4(*)	5.0	5.4
Total time (secs)				18128(*)	153595	21717	699(*)	52591	71554

Figure 2 shows our results on the solution quality in a comparative way for our real-case problem instances of large size. The gap between the result of a given method and the best known bound is shown. On the abcisse, Sxx represents the different graphs for the case of Sweden, Gxx represents DalGavle graphs, Dxx Dalarna graphs and dxx dalarna54 graphs. As shown in Table 2, some results are missing, due to unterminated solution processes. The table details the quality results for Sweden data, and also indicates the computational effort via the total runtime of the different algorithms. TIME and MEM indicate that the corresponding algorithm was not able to complete, either within a limited amount of time (2 days) or due to memory limitations (the software aborts before completion on our computer with 32 Gb of memory).

The quality deviation row of Table 2 shows the average deviation in percent between the solution found by the corresponding algorithm and the best lower bound. The best lower bound is the lower bound found either by CPlex or by the Volume algorithm. To compute this standard deviation, we only take into account the set of problem instances for which the algorithm terminates. This explains why algorithms which fail on large graphs (like CPlex) have suprisingly good values. The same explanation holds for global runtimes.

Among the algorithms that can handle all of the Swedish cases, imp-GA provides the best results on average, with a standard deviation of 5.0 %. It performs particularly well for small values of p. Concerning runtimes, as opposed to the Beasley "small" benchmark results, it outperforms the Volume algorithm with a total benchmark runtime of 52591 s vs 71554 s.

Memory utilization is a major concern for the CPlex algorithm which fails on some instances. For the Volume algorithm, almost 22 GB of memory are needed to work on graphs representing Sweden while Simulated Annealing and imp-GA use less than 2.8 GB for the largest cases.

8 Conclusion and Future Work

In this paper, we have compared the quality of several algorithms with respect to real-case instances of the p-median problem, which are large and whose demand is non-uniformly distributed. For this, we have introduced a new benchmark including up to 1938 candidates nodes. The CPlex approach is able to find optimal solutions for all of the Beasley testcases but fails to provide results for our set of very large graphs. Simulated annealing and the basic genetic approach exhibit average results. A hybrid genetic algorithm called hyper-GA has been tested. It improved the results of the basic GA on most of the graphs but failed on the largest graphs. The Volume algorithm has also been tested but its results vary in quality depending on the size of the problem instance.

To obtain better results for the large graphs, we have introduced a new hybrid genetic algorithm called `imp-GA`. This algorithm outperforms all other tested methods on large graphs and has a memory footprint which is as small as that of Simulated Annealing. Its runtime is lower than that of the Volume algorithm. Our results exhibit well the trade-off between exact and approximate methods in dependence on the size of the problem. The effect of a hybrid mutation within a genetic algorithm is important. Therefore, a good design of such a mutation, smartly restricting neighborhood search, leads to an efficient algorithm, especially for large problem instances. Since genetic algorithms can be efficiently parallelized, the proposed method `imp-GA` is a good candidate to deal with large real-case p-median problems.

In the future, we envisage to study other heuristics with respect to large problem instances. Observe, that in our approach distances have been precomputed from the graph. As an alternative we could design methods which extract an interesting set of candidate nodes from a dense graph, or which better exploit the planarity which is typical for graphs arising from geographical problems.

References

1. Al-Khedhairi, A.: Simulated annealing metaheuristic for solving p-median problem. Int. J. Contemp. Math. Sci. **3**(25–28), 1357–1365 (2008)
2. Alp, O., Erkut, E., Drezner, Z.: An efficient genetic algorithm for the p-median problem. Ann. Oper. Res. **122**(1–4), 21–42 (2003)
3. Ashayeri, J., Heuts, R., Tammel, B.: A modified simple heuristic for the p-median problem, with facilities design applications. Robot. Comput. Integr. Manufact. **21**(4–5), 451–464 (2005)
4. Avella, P., Boccia, M., Salerno, S., Vasilyev, I.: An aggregation heuristic for large scale p-median problem. Comput. Oper. Res. **39**(7), 1625–1632 (2012)
5. Avella, P., Sassano, A., Vasil'ev, I.: Computational study of large-scale p-median problems. Math. Program. **109**(1), 89–114 (2007)

6. Barahona, F., Anbil, R.: The volume algorithm: producing primal solutions with a subgradient method. Math. Program. **87**(3), 385–399 (2000)
7. Beasley, J.E.: OR-library: distributing test problems by electronic mail. J. Oper. Res. Soc. **41**(11), 1069–1072 (1990)
8. Cantú-Paz, E.: A survey of parallel genetic algorithms. Calculateurs Parallèles, Réseaux et Systèmes Répartis **10**, 141–171 (1998)
9. Carling, K., Han, M., Håkansson, J.: Does Euclidean distance work well when the p-median model is applied in rural areas? Ann. Oper. Res. **201**(1), 83–97 (2012)
10. Cleraux, C., Bourges, P.: Relaxation Lagrangienne et le problème du p-médian. Master's thesis, Institut Supérieur d'informatique, de modélisation et de leurs applications, Campus de Clermont-Ferrand/Les Cézeaux, BP 10125, 63173 Aubière CEDEX, France (2009)
11. Correa, E., Steiner, M., Freitas, A.A., Carieri, C.: A genetic algorithm for the P-median problem. In: Proceedings of 2001 Genetic and Evolutionary Computation Conference (GECCO-2001), pp. 1268–1275 (2001)
12. Corrêa, R., Ferreira, A., Rebreyend, P.: Scheduling multiprocessor tasks with genetic algorithms. IEEE Trans. Parallel Distrib. Syst. **10**(8), 825–837 (1999)
13. CPlex online reference manual
14. Gay, J.: Résolution du Problème du p-médian, Application à la Restructuration de Bases de Données Semi-Structurées. Ph.D. thesis, Université Blaise-Pascal, Clermont-II (2011)
15. Hakimi, S.L.: Optimum locations of switching centers and the absolute centers and medians of a graph. Oper. Res. **12**(3), 450–459 (1964)
16. Kariv, O., Hakimi, L.: An algorithmic approach to network location problems. SIAM J. Appl. Math. **37**(3), 539–560 (1979)
17. Lim, G.J., Ma, L.: Gpu-based parallel vertex substitution algorithm for the p-median problem. Comput. Ind. Eng. **64**(1), 381–388 (2013)
18. Lim, G.J., Reese, J., Holder, A.: Fast and robust techniques for the Euclidean p-median problem with uniform weights. Comput. Ind. Eng. **57**(3), 896–905 (2009)
19. Meng, X., Rebreyend, P.: From the road network database to a graph for localization purposes. Technical report 2014:09, Dalarna University, Statistics (2014)
20. Mladenoviç, N., Brimberg, J., Hansen, P., Moreno-Pérez, J.A.: The p-median problem: a survey of metaheuristic approaches. Eur. J. Oper. Res. **179**(3), 927–939 (2007)
21. Rebreyend, P., Han, M., Håkansson, J.: How do different algorithms work when applied on the different road networks when optimal location of facilities is searched for in rural areas? In: Huang, Z., Liu, C., He, J., Huang, G. (eds.) WISE Workshops 2013. LNCS, vol. 8182, pp. 284–291. Springer, Heidelberg (2014)
22. Reese, J.: Solution methods for the p-median problem: an annotated bibliography. Networks **48**(3), 125–142 (2006)
23. ReVelle, C.S., Swain, R.W.: Central facilities location. Geogr. Anal. **2**(1), 30–42 (1970)