Journal of Mathematics and System Science 5 (2015) 100-112
doi: 10.17265/2159-5291/2015.03.002

The P-Median Problem: A Tabu Search Approximation Proposal Applied to Districts

María Beatriz Bernábe Loranca, Rogelio González Velázquez and Martín Estrada Analco
Facultad de Ciencias de la Computación, Benemérita Universidad Autónoma de Puebla, Puebla, México

Received: December 30, 2014 / Accepted: January 26, 2015 / Published: March 25, 2015.

Abstract: P-median is one of the most important Location-Allocation problems. This problem determines the location of facilities and assigns demand points to them. The p-median problem can be established as a discrete problem in graph terms as: Let G = *(V, E)* be an undirected graph where *V* is the set of *n* vertices and *E* is the set of edges with an associated weight that can be the distance between the vertices $d_{ij}= d(v_i, v_j)$ for every *i, j* =1,…, *n* in accordance to the determined metric, with the distances a symmetric matrix is formed, finding $V_p \subseteq V$ such that $|V_p| \subseteq = p$, where *p* can be either variable or fixed, and the sum of the shortest distances from the vertices in $\{V\text{-}V_p\}$ to their closet vertex in V_p is reduced to the minimum. Under these conditions the P-median problem is a combinatory optimization problem that belongs to the NP-hard class and the approximation methods have been of great aid in recent years because of this. In this point, we have chosen data from OR-Library [1] and we have tested three algorithms that have given good results for geographical data (Simulated Annealing, Variable Neighborhood Search, Bioinspired Variable Neighborhood Search and a Tabu Search-VNS Hybrid (TS-VNS). However, the partitioning method PAM (Partitioning Around Medoids), that is modeled like the P-median, attained similar results along with TS-VNS but better results than the other metaheuristics for the OR-Library instances, in a favorable computing time, however for bigger instances that represent real states in Mexico, TS-VNS has surpassed PAM in time and quality in all instances. In this work we expose the behavior of these five different algorithms for the test matrices from OR-Library and real geographical data from Mexico. Furthermore, we made an analysis with the goal of explaining the quality of the results obtained to conclude that PAM behaves with efficiency for the OR-Library instances but is overcome by the hybrid when applied to real instances. On the other hand we have tested the 2 best algorithms (PAM and TS-VNS) with geographic data geographic from Jalisco, Queretaro and Nuevo León. In this point, as we said before, their performance was different than the OR-Library tests. The algorithm that attains the best results is TS-VNS.

Keywords: Metaheuristcs, P-mediana, PAM, Tabu search.

1. Introduction

In territorial partitioning, two models are the most common: the location-allocation and the set partitioning models. These models seek to group small geographical areas called basic units in a given number of bigger geographical clusters, named territories. The territorial partitioning problem can be modeled like a P-median problem with certain restrictions under the concept of partition, this is, if $\Omega = \{x_1, .., x_n\}$ is a finite set with *n* objects we wish to classify and let $k < n$ be the number of classes where we want to group the objects. A partition $P = (C_1, .., C_k)$ from Ω in k classes $C_1, .., C_k$, is characterized by the following conditions:

$$1.\ \Omega = \bigcup_{i=1}^{K} C_i$$

$$2.\ C_i \cap C_j = \emptyset, \text{for every } i \neq j$$

The P-median problem consists in finding the best configuration of facilities to attend the population's demand in the best way [2]. Given a set of *m* nodes (coordinates) in the Cartesian plane, where every node possesses a certain demand that must be fulfilled, *k* < *m* service providers must be installed to satisfy this demand at the minimum cost. The cost can be

Corresponding author: María Beatriz Bernábe Loranca, Ph.D. in Operative Research, research field: combinatorial optimization. E-mail: beatriz.bernabe@gmail.com.

determined in a proportional way to the distance between nodes.

The formulation to solve it was established in [3]. The problem was named p-median. This problem locates p facilities to minimize the total distance between the demand points and their nearest facilities [2]. For this we solve the allocation problem and minimize the distance and demand of the nodes. The computational complexity of this kind of problems requires using approximate methodologies to give a satisfactory response in regard to quality and time. In this point, two kinds of data have been processed with different heuristic variants based on the P-median model.

In this work, after testing different metaheuristic methods with OR-Library instances,we conclude that TS-VNS is the best method.For this reason, we chose TS-VNS to partition maps of different states of Mexico and we tested the efficiency of TS-VNS for the geographical data (electoral sections).

2. Materials and Methods

2.1 Transformation of the P-Median Combinatory Optimization Model to a Binary Integer Programming Model

In order to implement an approximation algorithm to search for solutions for the NP-hard problems it's necessary to approach the p-median problem (PMP) as a combinatory optimization problem (COP) given that the PMP is NP-hard [4] and usually represented as a binary integer programming problem (BIPP).

2.1.1 P-Median COP Model

We model the PMP as follows: given a set of n vertices of a graph in the plane denoted by $V=\{1, 2,..,n\}$, $|V| = n$, the goal is to find a subset of vertices $L \subseteq V$, with $|L| = p$ of possible locations for the medians such that the average total distance of the design is minimum. We say that this approach is of the COP kind because the feasibility space Ω from the PMP is formed by all the subsets of L of cardinality p of a set with cardinality n, with $p<n$. With this it is established as commented by [4].

$$|\Omega| = \frac{n!}{p!\,(n-p)!} = \binom{n}{p}$$

Overall an instance for the PMP is denoted by **PPM (*n, D, p*)** where n and p are as we said previously and $D=(d_{ij})$ is the distances matrix between all the pairs of vertices from V.

2.1.2 BIPP model for the p-Median problem

The PMP over a graph can be approached as a binary integer programming problem [5] in the following way:

$$\text{Let } x_{ij} = \begin{cases} 1 & \text{If vertex } j \text{ is assiged to vertex } i \\ 0 & \text{in any other case} \end{cases}$$

$$\text{Let } Y_i = \begin{cases} 1 & \text{if vertex } i \text{ is a median} \\ 0 & \text{in any other case} \end{cases}$$

$$Min\ Z \sum_{i=1}^{k} \sum_{j=1}^{n} d_{ij} x_{ij} \quad (1)$$

$$\text{subject to } \sum_{i=1}^{k} x_{ij} = 1\ \forall\ j = 1, \dots, n \quad (2)$$

$$\sum_{j=1}^{n} x_{ij} \le n\, y_i\ \ \forall\ i = 1, \dots, k \quad (3)$$

$$\sum_{i=1}^{k} y_i = p \quad (4)$$

Where equation (1) is the objective function, the decision variables x_{ij} and y_i are binary and where k is the potential number of vertices where usually a median can be located $k=n$, p is the fixed number of required medians. The restrictions (2) guarantee that each vertex has an associated median, the restrictions (3) determines the distribution of the vertices to the medians and (4), the number of medians.

2.1.3 A Case for the transformation.

The PMP as a BIPP has a feasible space of the exponential 2^n kind and as a COP is $\binom{n}{p}$

Then in defining a transformation $T: 2^n \to \binom{n}{p}$

We consider a case of a graph of n = 17 vertices, this is V= SDAAF illustrated in Fig. 1 which solution was determined by the BIPP model illustrated in Fig. 2 giving the following results with a cost C given by the objective function (1).

The solution is shown in Table 1, in the first binary solution it determines the medians in

$$y_6 = 1, y_7 = 1, y_{15} = 1, y_{13} = 1$$

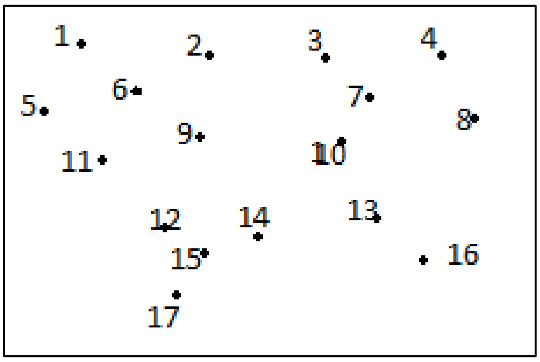

Fig. 1 17 vertices graph.

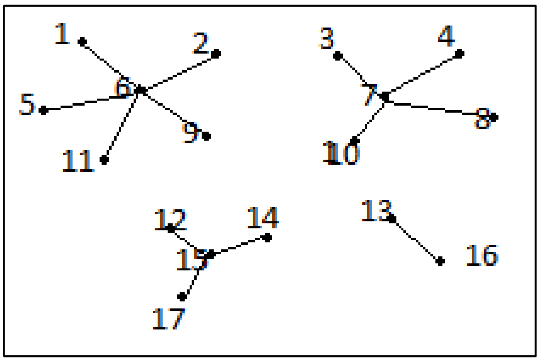

Fig. 2 $n = 17$ with $p = 4$ case.

Table 1 Binary Solution BIPP.

Binary solution	Median	Group
$x_{61} = 1, x_{62} = 1, x_{65} = 1, x_{69} = 1, x_{611} = 1$	6	6, 1, 2, 5, 9, 11
$x_{73} = 1, x_{74} = 1, x_{78} = 1, x_{710} = 1$	7	7, 3, 4, 8, 10
$x_{1512} = 1, x_{1514} = 1, x_{1517} = 1$	15	15, 12, 14, 17
$x_{1316} = 1$	13	13, 16

With its consequent combinatory solution L={6, 7, 13, 15} that has a cost C obtained from the sum of the sums of the distances between the medians and their associated vertices, this is C = *d*(6,1)+d(6,2)+*d*(6,5)+ *d*(6,9)+ *d*(6,11)+ *d*(7,3)+ *d*(7,4)+ *d*(7,8)+ *d*(7,10)+ *d*(15,12)+ *d*(15,14)+ *d*(15,17)+ *d*(13,16).

The test instance for the combinatory problem is **PPM (17, D_{17x17}, 4)** and it shows the equivalence between the COP and BIP solutions for the PMP. The same interpretation from a matrix approach placing the binary variables in a matrix X_{ij} and from the intersections between rows and columns we can observe how the groups form a partition of V, we can deduce that the matrix from Table 2 is equivalent to the combinatory solution discussed.

2.2 Algorithms

In this section we present the algorithms developed. Simulated Annealing (SA) and Variable Neighborhood Search (VNS) have been widely published and tested for the data we present here, however, considering that VNS has attained good results [6], we have insisted on broadening the basic VNS proposal by incorporation two variants: a bioinspired VNS [7] and a Hybrid Tabu Search with some principles from VNS.

2.2.1 Extension to VNS: VNS BIO and TS-VNS

In this section we focus on the main part of VNS where the changes in the neighborhood structures are fundamental.

The Neighborhood Search procedures traverse the solutions space U employing a set of transformations or moves. The solutions that are obtained from another one through one of the possible moves are known as the neighbors of this solution and constitute its neighborhood. The set of possible moves gives place to a neighborhood relationship and a neighborhood structure in the solutions space. The general scheme of a neighborhood search procedure consists in generating an initial solution and, until a stopping criterion is met, a move is iteratively selected to modify the solution [8, 9].

The neighborhood of a solution is formed by the solutions that can be accessed from it by one of the possible moves.

Formally, a neighborhood structure over a space or search universe U is a function E: U $\rightarrow 2^S$ that associates a neighborhood E(x) $\subseteq$ U to every solution x of U. A big amount of heuristic methods proposed in the literature belong to the neighborhood search procedures class.

Table 2 BIPP Matrix.

ij	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1																	
2																	
3																	
4																	
5																	
6	1	1			1	1			1		1						
7			1	1			1	1		1							
8																	
9																	
10																	
11																	
12																	
13													1			1	
14																	
15												1		1	1		1
16																	
17																	

The description in pseudocode of the neighborhood search is shown below:

Algorithm 1. VNS Algorithm

```
Procedure neighborhood search
{
x ← generate Solution (U)
x* ← x
Do {
x ← select solution (E(x));
If (object (x) improves object (x*))
x* ← x;
}while (not stop criterion)
}
```

The selection of the neighborhood structure is fundamental in the success of the search procedures since it determines the quality of the moves set applied. The combined moves appear when several moves are executed subsequently over a solution. An adequate combination of moves enriches the neighborhoods, which allows taking wider steps in the approaching to the optimum. An important characteristic of the moves is the feasibility of the contributed solutions.

Formally, the procedures that only take into account feasible moves are associated to the concept, somewhat more restrictive, of neighborhood structure as a function $E: S \rightarrow 2^S$ that associates a neighborhood $E(x) \subset S$ of feasible solutions to each feasible solution x of S.

The main neighborhood search metaheuristic focuses on the move selection procedure only. However, besides the selection of a neighborhood structure over which to articulate the search; there are other relevant questions regarding the success of the neighborhood search procedure, such as: the objective function evaluation, the procedure to generate the initial solution and the stopping criterion.

2.2.1.1 Bioinspired VNS

This VNS algorithm is inspired in the human behavior in emergency or social instability situations. Specifically this analogy is found in the local search procedure of this algorithm. After the initial solution is generated the local search will make small changes over the solution imitating a social behavior as follows: Every country has certain a power structure ideally formed for the interests of the people, each entity of the country has people's representative, however as we have seen throughout history, the social dissatisfaction destabilizes these groups and in extreme cases the represented change their representatives, however when the problem is in a smaller scale each person is free to join the political party of its choice or a group or committee formed by people with similar interests and ideologies. Our proposal covers both cases; the furthest objects (less similar) from the medoid (representative) are moved to another group that is close, emulating social grouping, but when these small changes reach an established limit, the neighborhood structure change occurs emulating a big scale social conflict by restructuring one or more groups, replacing their representatives and reorganizing the groups accordingly, looking to reduce the differences or distances between the objects and the medoids. This solution restructuration will happen every time the neighborhood structure changes and after each change the local search will continue over the new solution according to the input parameter given by the user.

In this algorithm the intensification strategy around good solutions is handled by the local search which remains in a single structure in an attempt to reach difficult optimal solutions around the neighborhood. On the other hand the diversification comes into action when the current neighborhood has been exhaustively explored and therefor it's necessary to proceed to another area of the solutions space, this is made by the systematic neighborhood structure change N_1, N_2, …,N_k that we have defined already. This is the way the algorithm achieves the balance necessary to escape from local optima and reach deeper zones in the solution space.

Algorithm 2.

1. Load dissimilarity matrix (*n x n*)
2. Read VNS parameters: VNS iterations and local search iterations *itVNS, itBL*
3. Read number of groups to form - *k*
4. Initialize geographic objects array *objArray*
5. Initialize VNS metaheuristic with the parameters given by *itVNS* and *itBL*
6. glo_cont = 1
7. ls_cont = 1
8. solFound = False
9. build and initial soltuion
10. While ls_cont<*itVNS*
    a. Nk = 1
    b. WhileNk< n.
    Local Search:
    (1) Move to the Nk neighborhood structure of the current solution
    (2) While *cont<itBL* AND solFound = False
    a. Select a centroid, which represents the group with the biggest dissimilarity (*c1)*
    b. Select an element (*obj*) assigned to c1 which distance to c1 is the largest among the rest of the elements in the group.
    c. Select a centroid *(c2)* that contains the non-centroid object closest to *obj*.
    d. The object *obj* is reassigned to *c2* and the solution cost is modified according to the reallocation made.
    e. Forbid the selection of *c1*and the object *obj* for a certain amount of time (Tabú)
    f. If the cost of the new local solution (*costeSp*) is bigger than the cost of the current solution *solFound* = True else *cont = cont + 1*
    (3) End while
    (4) Nk = Nk + 1
    c. End while
11. End while

The algorithm reads a dissimilarity matrix, then it reads the VNS parameters; itBL and itVNS, the first one indicates the number of local search iterations that will be performed inside each neighborhood structure, the algorithm is designed to generate *n* neighborhood structures and the itVNS parameter tells us how many times all of them will be explored. We have an array of objects which distances are obtained from the dissimilarity matrix. So the algorithm repeats its main cycle itVNS times, this cycle generates the N_k neighborhood structures from a randomly generated initial solution. Over this solution and over each successive neighborhood structure, the local search that begins in line 10.b.i will be executed and will finish when the value itBL is reached or when a worse solution is found. The local search begins by finding the medoid from the least similar (biggest distances) group, after this the algorithm finds the object that is furthest from the medoid, finally the object is reassigned to another group that contains the closest or most similar object to itself and the solution cost is updated accordingly. In this step is necessary to implement a small tabu strategy to avoid repeatedly choosing the same object and group.

2.2.1.2 Tabu Search and VNS Hybrid

From our experience with our previous algorithms, we built a hybrid with the two best candidates: VNS and TS. Our objective is to achieve results comparable to the quality of PAM that has shown great efficiency to reach optimal solutions for several tests, however is very slow for big-sized problems, for this reason with our algorithm we also seek to maintain a certain speed to deal with a bigger range of problems.

To fulfill these goals we needed to explore new strategies. For example, for the initial solution we chose the Stingy Drop or Greedy Drop method [9], this greedy is capable of forming a good compact initial solution which works well for up to 1000 - 1500 objects, after that limit the performance of the algorithm drops significantly (see section 3). This method assigns all the available objects as medoids or P's and they are eliminated one by one until the desired number of P's is reached. The elimination process drops the medoid that would provide the biggest gain if removed.

The second most important part of any metaheuristic is the neighborhood function, for this proposal we have implemented a variation of the well-known interchange or swap method proposed by Whitaker in [10].

Taking into account these strategies we developed the following hybrid algorithm:

Algorithm 3. TS-VNS Hybrid

```
Input:
Number of facilities - p
Number of iterations - nit
Number of iterations for second phase - nit2
Number of worse solutions permitted (perturbation) -ip
Tabu Tenure -tt
pc ← 0
ic ← 1
S ←InitialSolution()
S* ← S
While ic < nit do
prev_cost ← Cost(S)
Move(S)
If Cost(S) > prev_cost then
pc ← pc+1
end if
If Cost(S) < Cost(S*) then
S* ← S
end if
If pc > ip then
ChangeNeighborhood(S)
pc ← 0
end if
UpdateTabuLists()
ic← ic+1
end while
S ← S*
CleanTabuStates()
For I ← 0 until nit2 do
Move(S)
If Cost(S) ← Cost(S*) then
S*←S
end if
UpdateTabuLists()
ic ← ic+1
end if
Return S*
```

As we can see this algorithm has a classic TS structure but a strategy to change neighborhood structures was added, which we define as the replacement of all the medoids except one (determined by a counter that will be increased by 1 after each structure change, this value represents a number of object, starting from 1 to cover all the input objects).

The neighborhood structure change occurs as a reaction to a quality limit established by the user; this is when a certain number of worse solutions have been obtained in the current structure.

The input parameters are *p* (number of facilities), *nit* (global iterations), *nit2* (second phase iterations), *ip*(number of worse solutions allowed in the current neighborhood) and *tt* (tabu tenure for the dropped and added medoids) which indicates the time that the elements exchanged in the local search won't be able to move from their current position/state.

Our local search algorithm based on the swap method is described below:

Algorithm 4. Modified Swap Method.

Input:

Array of non-tabu facilities - medoids

Array of non-tabu objects - ugs

1: index ← RandomIndexFrom(medoids)

2: **If** Size(medoids[index]) = 0)

3: i_ug ← RandomIndexFrom(Size(ugs))

4: new_medoid ← ugs[i_ug]

5: removed_medoid←medoides[index]

6: **else**

7: new_medoid ← c_matrix[0][index]

8: local_cost←*findOut*(new_medoid)

9: f_rem ← local_cost[1]

10: profit ← local_cost[2];

11: **For** i **from** 1 **to**Size(medoids[index]) **do**

12: t_medoid ← c_matrix[i][index]

13: t_cost ← *findOut*(t_medoid)

14: **If** t_cost[2] > profit **then**

15: new_medoid ← t_medoid

16: f_rem ← t_cost[1]

17: profit ← t_cost[2]

18: **end if**

19: **end for**

20: index ← f_rem

21: removed_medoid← medoides[index]

22: **end if**

23: Remove(medoides[index]

24: Remove(ugs[new_medoid])

25: TabuAddStart(new_medoid)

26: TabuDropStart(removed_medoid)

27: *Update Operations*

The local search will remove a facility from the array of medoids and will choose a non-facility object to replace the one dropped. Our improvement to Whitaker's method consists in using a semi-random strategy that evaluates only a limited candidate list because the computational cost of evaluating all the possible swaps would be high.

First, our algorithm selects a random facility from the array of medoids, which, if it doesn't have any elements assigned then a random object from the array of objects ugs is chosen as a new medoid, then dropped facility is "tagged" as removed medoid. On the other hand if the medoid has dependent objects then this set of objects becomes the candidate list to evaluate. The matrix c_matrix is the clusters matrix that stores the objects assigned to each medoid, this allows an O(1) access to them. In line 11 the evaluation cycle begins using the *findOut* function which is defined in [10]. This function reads a candidate object and returns two values; a medoid that if swapped by the input object would return the biggest gain, and the value of this gain, there for local_cost and t_cost are arrays of size 2. Finally the medoid stored in removed_medoid and the object stored in new_medoid are dropped from the medoids array and the ugs array respectively and their tabu-active state starts. If this wasn't a tabu search method the last part of the procedure would be a simple swap operation that would put new_medoid in medoids and removed_medoid in ugs. The update operations are a set of operations that updates the cost of the solution, the clusters matrix and as suggested in [10], the auxiliary pre-calculated gain and loss arrays.

2.3 Test Instances

We have selected two sets of instances; uncapacitated p-median instances from OR-Library [1] and real geographical data from three states of Mexico converted from the data available in [11].

All of the algorithms were run in a windows pc (CPU: AMD E-350 at 1.60 GHz, RAM: 2GB DDR2).

3. Results and Discussion

The following Tables contains the tests for the OR-Library instances.

The details of each one of the 40 instances can be seen in [1], they go from 100 to 900 objects and the values of P, from 5 to 200. The values in bold in both Tables represent the tests that returned the best known value (probably the global optimum) until today.

We can see that PAM achieves the best results but its computing time increases as the problem size increases. The worst algorithm was SA.

Even though TS-VNS attains good results for the instances in Table 4 it's not a guarantee that it will be the same for the real geographical data of our interest. For this reason we selected PAM and TS-VNS to test them with data from three states of Mexico: Jalisco, Querétaro and Nuevo León (3484, 814 and 2416). Our results are in Table 5.

For Table 5 we executed 9 instances, 3 for each map using 12, 24 and 48 P's for each.The instances 1,2 and 3 are for the map of Querétaro that has 814 objects, instances 4, 5 and 6 are for Nuevo León with 2416 objects and 7, 8 and 9 are the instances for Jalisco which has 3484 objects.

We see in Table 5 that TS-VNS surpasses PAM in all the instances except for instance 3 (Querétaro with P = 48), however there's a big difference between the execution times. A peculiar aspect is that TS-VNS was run with the same input parameters for all the instances: nit = 1000, nit2 = 500, ip = 20 and tt = p/2 (see section 2.2.1.2 for more details) but its execution time considerably decreases, this is because of our modified swap method where the candidate list is formed by the objects assigned to the selected medoid, therefor when the P is bigger the objects are more evenly distributed, generating smaller candidate lists for each medoid. For example for the map of Jalisco with P = 12 the algorithm finished in 15 minutes and for P = 48, in almost 6 minutes. Another aspect to consider is that the greedy method to generate the initial solution was only used for the map of Querétaro, because it required several minutes to generate the solution for the other maps, for this reason we used a randomly generated solution for Jalisco and Nuevo León but as we can see this didn't negatively affect the quality of the final solution, as we still obtained better results than PAM.

These maps were generated with a customGeographic Information Software developed in Java with the aid of GeoTools library, available for free in [12].

4. Conclusions

We observed that PAM surpassed the quality of all the other algorithms in the P-median instances from OR-Library. Even when we tried to give TS-VNS more time to find a solution, in many cases didn't manage to match the quality of PAM, not even do better than PAM in the instances where neither matched the best known result, however, as we saw in Table 5 TS-VNS worked better than PAM, in quality and time, when working with geographical data, this is an issue that we need to examine in more detail to find the reasons for this behavior and make the necessary changes to achieve a consistent behavior with different kinds of data. For this we need to do more testing and debugging with our TS-VNS algorithm.

Table 3 Results for the OR-Library P-Median instances (Part 1).

Instance	VNS		SA		VNS-BIO	
	Cost	T	Cost	T	Cost	T
1	**5819**	103	6209	28	5884	0.55
2	4341	180	4646	70	4601	7.872
3	4467	180	4785	47	4870	7.793
4	3380	250	3693	93	3744	2.877
5	1664	360	1820	119	1771	2.749
6	7917	330	8349	49	8236	2.427
7	5952	540	6446	104	6415	3.1
8	5204	1003	5536	174	5507	2.647
9	3385	1860	3626	286	3432	2.134
10	1700	1980	1850	907	1844	2.662
11	7803	720	8346	149	7968	9.762
12	7200	900	7717	298	7485	9.615
13	5126	1860	5475	692	5444	9.877
14	3823	1440	3992	71	3907	10.403
15	2464	240	2558	1721	2482	10.354
16	8423	300	8958	220	8736	21.732
17	7651	540	8197	322	7905	20.896
18	5821	2700	6038	505	5935	22.986
19	3747	2760	3881	2036	3780	24.238
20	2647	2040	2755	1909	2707	24.865
21	9557	240	10231	102	9794	43.587
22	9433	300	9802	170	9744	38.277
23	5645	3600	5941	839	5827	44.257
24	3974	600	4065	1440	3988	46.955
25	2726	720	2852	240	2810	51.757
26	10312	60	10869	14	10458	64.285
27	9065	120	9511	25	9159	80.152
28	5664	480	5799	141	5768	75.77
29	4114	780	4176	70	4132	86.531
30	2960	1320	3058	47	2996	122.48
31	10528	70	11157	93	10739	105.61
32	10383	120	10818	119	10606	34.87
33	6007	720	6166	49	6102	43.789
34	4193	1500	4286	104	4191	53.045
35	11037	120	11698	174	11180	47.757
36	9994	180	11544	250	11154	60.261
37	6460	1620	6715	50	6602	62.387
38	11725	180	12252	10	11678	68.799
39	10570	300	11017	25	10599	62.698
40	6632	2460	6803	40	6664	72.65

Table 4 Results for the OR-Library P-Median instances (Part 2).

Instance	PAM		VNS-TS	
	Cost	T	Cost	T
1	**5819**	0	**5819**	2.556
2	4105	0	**4093**	1.672
3	**4250**	0	**4250**	1.604
4	3046	1	3041	5.703
5	**1355**	1	1394	5.928
6	**7824**	0	**7824**	49.28
7	5645	1	**5631**	21.744
8	4457	2	4451	19.764
9	2753	8	2804	31.729
10	1263	14	1318	25.288
11	**7696**	0	**7696**	145.137
12	**6634**	1	**6634**	63.67
13	**4374**	20	4388	48.169
14	2974	56	3091	37.845
15	1738	82	1858	48.857
16	**8162**	1	**8162**	222.629
17	**6999**	2	**6999**	97.449
18	4811	67	4840	25.538
19	2859	296	2927	29.422
20	1805	600	1882	36.45
21	**9138**	0	**9138**	164.141
22	8669	4	8579	58.606
23	**4619**	160	4664	58.606
24	2965	938	3093	127.046
25	1844	1608	1937	132.722
26	**9917**	2	**9917**	389.316
27	**8307**	9	**8307**	68.365
28	4515	605	4551	35.594
29	3039	2101	3181	66.12
30	2009	2208	2119	105.318
31	**10086**	2	**10086**	479.083
32	**9301**	8	**9310**	109.158
33	4703	1495	4735	47.558
34	3026	4685	3168	119.309
35	**10400**	2	**10400**	413.429
36	**9934**	10	**9934**	141.098
37	5064	2092	5278	68.316
38	**11060**	8	**11060**	86.544
39	**9423**	13	**9423**	99.102
40	5138	5076	5214	76.852

Table 5 Results for the geographical data.

Inst.	TS-VNS		PAM	
	Cost	Time	Cost	Time
1	**50.7595**	00:00:56	51.59754	00:02:03
2	**33.9236**	00:00:35	33.93228	00:15:33
3	23.7555	00:00:36	**23.338**	00:25:45
4	**210.9751**	00:08:05	211.2497	00:15:38
5	**139.5007**	00:04:21	140.1448	01:49:55
6	**94.5667**	00:02:15	**Didn't finish after 5 hours**	
7	**529.9198**	00:15:01	531.7125	00:24:28
8	**371.3131**	00:09:58	**Didn't finish after 5 hours**	
9	**243.5297**	00:05:51	**Didn't finish after 5 hours**	

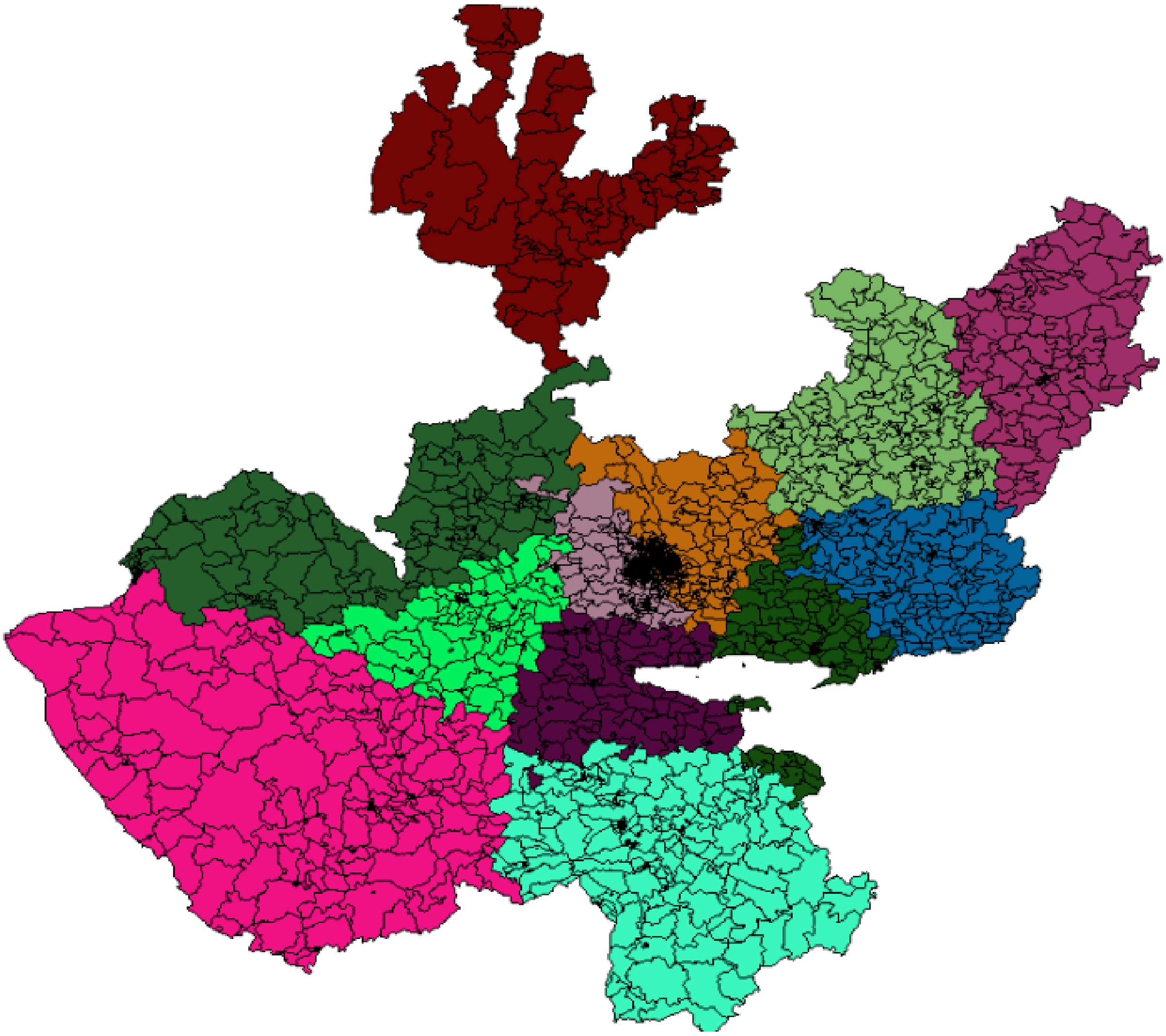

Fig. 3 Map of Jalisco with P = 12 returned by PAM. Cost: 531.7125.

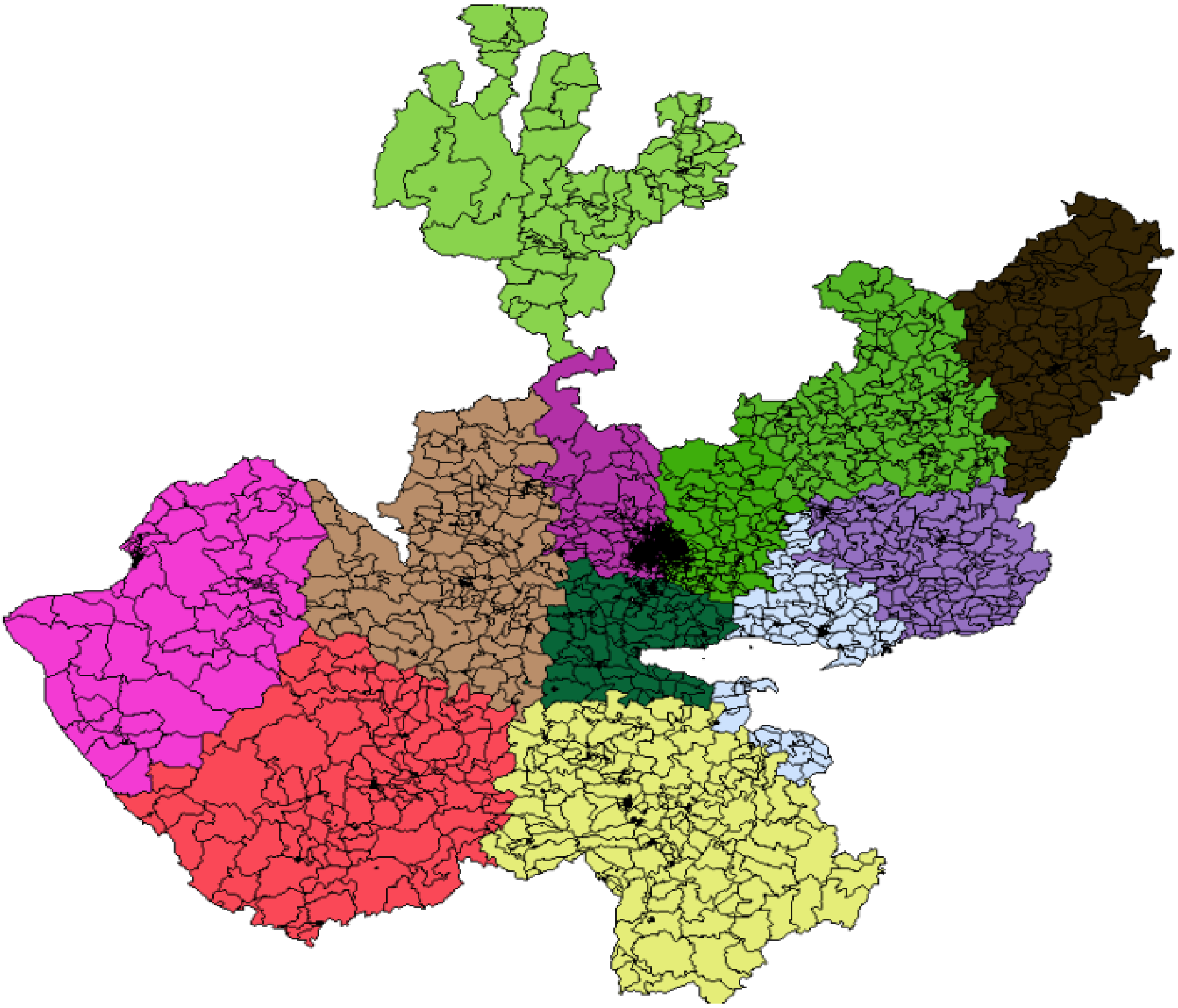

Fig. 4 Map of Jalisco with P = 12 returned by TS-VNS. Cost: 529.9198.

References

[1] OR-library, Distributing test problems by electronic mail. Journal of the Operational Research Society, 41, 1069-1072 (1990)

[2] http://people.brunel.ac.uk/~mastjjb/jeb/orlib/pmedinfo.html Retrieved on August 25, 2014.

[3] M. S. Daskin, Network and Discrete Location: Models, Algorithms and Applications , John Wiley and Sons, Inc., New York (1995).

[4] S. Hakimi, Optimum location of switching centers and the absolute centers and medians of a graph. Operations Research, 12, (1964), pp. 450-459.

[5] O. Kariv, S. L. Hakimi, An algorithmic approach to network location problems" , Part II, The p-Medians, SIAM Journal of Applied Mathematics 37, (1979), pp. 539-560.

[6] R. L. Church, COBRA: A New Formulation of the Classic p-Median Location Problem, Annals of Operations Research 122, (2003), pp. 103–120.

[7] M. B. Bernábe, J. E. Espinosa, J. Ramirez, M. A. Osorio, A Statistical Comparative Analysis of Simulated Annealing and Variable Neighborhood Search for the Geographical Clustering Problem, Computación y Sistemas, ISSN 1405-5546, Vol. 14 No. 3, (2011)pp. 295-308.

[8] M. B. Bernábe, R. González, E. Olivares, J. Ramírez and M. Estrada, A Bioinspired Proposal of Clustering Around Medoids with Variable Neighborhood Structures, International Journal of Computer Information Systems and Industrial Management Applications. ISSN 2150-7988, Vol. 6 (2014) pp. 45 – 54 © MIR Labs, www.mirlabs.net/ijcisim/index.html, Dynamic Publishers, Inc., USA.

[9] N. Mladenovic and P. Hansen, Variable Neighborhood Search, Computers & Operations Research, Vol 24 No. 11, (1997)pp.1097-1100.

[10] N. Mladenovic, J. Brimberg, Hansen, P., Moreno, J. A.:

The p-median problem: A survey of metaheuristic approaches. European Journal Operational Research, Vol. 179 (2007).

[11] M. Resende, R. Werneck, A fast swap-based local search procedure for location problems. Annals of Operational Research Vol. 150, (2007), pp. 205-230.

[12] INEGI, Sistema Estadísticas Censales a Escalas Geoelectorales, (2010), http://gaia.inegi.org.mx/ geoelectoral/viewer.html# retrievedonDecember 21, 2014.

[13] GeoTools, GIS utilities library for Java, http://www.geotools.org/, retrieved on December 21, 2014.