A Genetic Algorithm for Solving the P-Median Problem

[*]Abdel Latif Abu Dalhoum, [*]Al Zoubi, Moh'd, [**]Marina de la Cruz, [**]Alfonso Ortega, [**]Manuel Alfonseca
*King Abdullah II School for Information Technology, University of Jordan
Email: {a.latif, mba}@ju.edu.jo
**Escuela Politécnica Superior, Universidad Autónoma de Madrid
Email: {Marina.Cruz, Alfonso.Ortega, Manuel.Alfonseca}@uam.es

Keywords: location allocation problem, p-median model, grammar evolution, Christensen grammar.

ABSTRACT

One of the most popular location-allocation models among researchers is the p-median. Most of the algorithmic research on these models has been devoted to developing heuristic solution procedures. The major drawback of heuristic methods is that the time required finding solutions can become unmanageable. In this paper, we propose a new algorithm, using different variants of grammar evolution, to solve the p-median problem.

Acknowledgement: This work has been sponsored by the Spanish Ministry of Science and Technology (MCYT), project number TIC2002-01948.

1. Introduction

The field of location-allocation modelling has been widely used in different application areas. It essentially consists of a set of techniques for determining "good" locations of facilities for providing goods and services, including industrial facilities, schools, warehouses, fire stations, voting centres and fast-food restaurants (see [Daskin 1995] for application details).

The p-median model is perhaps the most popular location-allocation model among researchers. It has been shown that a large number of location problems can be solved by p-median [Densham and Rushton 1992], [Hilsman 1984]. The goal of the p-median model is to select the locations of p facilities to serve n demand points, so that the sum of the distances between each demand point and its nearest facility is minimized. As an optimization problem, the p-median model is shown to be NP-hard [Megiddo and Supowit 1984]. Therefore, most of the algorithmic research on this problem has been devoted to developing heuristic solution procedures.

One of the oldest and most frequently used heuristics to solve the p-median problem is the exchange algorithm, originally proposed by [Teitz and Bart 1968] and later used by many other authors [Garcia-Lopez et al 2002], [Hansen et al 2001], [Hodgson 1978], [Resende and Werneck 2003], [Whitaker 1983]. The algorithm starts with an initial solution to the initial facility set, and the p-median objective function is computed for this solution. The algorithm then seeks to improve the initial solution by exchanging points of the facility set for points of the non facility set. The objective function is computed after each exchange. Swaps are allowed only if the objective value is improved. The algorithm stops when no improvement in the objective value can be found. The major drawback of the exchange algorithm is that the time required for finding solutions can become unmanageable; therefore, the algorithm is especially successful when small problems are involved [Densham and Rushton 1992].

In this paper, we have applied two versions of grammar evolution [O'Neill and Conor 2003] to solve the p-median problem. Section 2 in the paper describes the p-median model in some more detail. Section 3 summarizes the field of grammar evolution. Section 4 describes our use of standard grammar evolution to solve the p-median problem. In section 5, we use instead Christiansen grammar evolution [Christiansen 1990] [Shutt 1993] for the same purpose. Finally, section 6 provides the conclusions we have reached and our future research objectives.

2. The p-median model

The p-median model is a solution to the location-allocation problem, which locates p facilities among n demand points and allocates each demand point to one of the facilities. The objective is to minimize the sum of the distances between each demand point and its nearest facility. The p facilities composing a solution for the problem are called the medians.

Assuming that every demand point can be elected as a median, the p-median problem is formally stated as follows [ReVelle and Swain 1970]:

$$\min = \sum_{i=1}^{N}\sum_{j=1}^{n} w_i d_{ij} x_{ij} \qquad (1)$$

subject to:

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad \forall i, \qquad (2)$$

$$x_{ij} \le y_j \qquad \forall i, j, \qquad (3)$$

$$\sum_{j=1}^{n} y_j = p, \qquad (4)$$

$$x_{ij} = 0 \text{ or } 1 \qquad \forall i, j, \qquad (5)$$

$$y_i = 0 \text{ or } 1 \qquad \forall j, \qquad (6)$$

where:

N	total number of demand points,
n	Total number of facilities,
x_{ij}	1 if point i is assigned to facility located at point j, 0 otherwise,
y_i	1 if point j is allocated as a facility, 0 otherwise,
w_i	demand at point i,
d_{ij}	distance from point i to point j,
p	desired number of facilities to be located.

The objective function (1) to be minimized the sum of the (weighted) distances between the demand points and the medians. Constraint (2) guarantees that all the demand points are assigned to exactly one facility location. Constraint (3) forbids that a demand point be assigned to a facility that was not selected as a median. Constraint (4) defines the total number of medians as p. Constraints (5) and (6) guarantee that the values of x and y are binary (0 or 1).

3. Grammar evolution

Genetic algorithms are optimization tools that simulate the principles of natural evolution and search for the minimum of an objective function.

Genetic algorithms operate on a population of chromosomes. An objective function (termed fitness function) provides the mechanism to evaluate each element in the population. From the current population, the next population is generated using several probabilistic operators: selection, crossover, mutation, elision and fusion.

Selection: The best elements in the population are selected according to their fitness. Solutions with the best fitness have better chance to survive in the next generation.

Crossover: the crossover operator combines the genotypes of two elements to generate new progeny, exchanging parts of the parental genotypes.

Mutation: some components in the progeny genotypes are modified randomly.

Elision: some components in the progeny genotype are randomly deleted.

Fusion: some components in the progeny genotype are randomly replicated.

Grammar evolution (GE) [O'Neill and Conor 2003] is an automatic evolutionary programming algorithm based on strings, independent of the language used. Genotypes are represented by strings of integers (each of which is named *codon*) and the grammar of the target programming language is used to deterministically map each genotype into a syntactically correct phenotype (a program). This allows GE to avoid one of the main difficulties in automatic evolutionary programming [Koza 1992].

The following scheme shows the way in which GE combines traditional genetic algorithms with genotype to phenotype mapping:

1) Generate at random an initial population of genotypes.
2) Translate each member of this initial set into its phenotype.
3) Sort the genotype population by their fitness (computed from the phenotypes).
4) If the best individual is a solution, the process ends.
5) Replace the worst individuals by the genetically modified offspring of the best individuals.
6) Go to step 2.

GE genotypes are deterministically translated by applying to each codon the following process:

1) Choose the leftmost non terminal symbol in the current word.
2) Number the n right hand sides of all the rules for this non-terminal symbol (from 0 to n-1).
3) Select the right hand side of the rule whose number equals *codon mod (number of right hand sides for this non-terminal)*
4) Derive the next word by replacing the non terminal by the selected right hand side.

4. Christiansen Grammar evolution

Christiansen grammars [Christiansen 1990] [Shutt 1993] are an extension of attribute grammars, where the first attribute associated to every symbol is a Christiansen grammar. Derivation relationship is redefined to make the model adaptable: the first attribute contains the rules applicable to the corresponding symbol. As with any other attribute, its value can be computed while the grammar is being used, thus the grammar may be changed on the fly.

Several formal notations have been used to describe Christiansen grammars. This paper follows that used in [Shutt 1993], which is very similar to typical attribute grammars. It is slightly more declarative, and explicitly specifies, for every attribute, whether it is inherited (↓) or synthesized (↑). The full syntax is as follows:

- Non terminals are followed by the list of their attributes.

- In the production rules, the names of the attributes are implicit. Their values are used instead.
- Additional semantic actions, which cannot be expressed by the values of the attributes, follow the rule between brackets. These actions are usually written in pseudo code.

Christiansen grammar evolution makes the GE genotype to phenotype mapping adaptive, by using a Christiansen grammar in place of the context-free grammar normally used in GE. The Christiansen grammar is designed to express both the syntactic and the semantic conditions that a valid phenotype must comply with.

CGE adds the following tasks to the previous algorithm:

1.1) Evaluate the attributes

1.2) Select the applicable rules from the first attribute in each non terminal

Our algorithm borrows a few interesting theoretical results from syntactic analysis techniques. Reference [Aho et al 1986] shows that syntactically driven left-to-right translation schemes guarantee the proper evaluation of the kind of attributes previously described. The same reference also shows that this kind of attributes can be considered complete (they can represent any kind of attributes) and are compatible with a left to right depth-first route across the derivation tree. Since the genotype to phenotype mapping builds trees to derive words, rather than to analyze them, backtracking is needed to ensure the proper conclusion of the translation.

Notice that the main feature of Christiansen Grammars is the modification of the set of rules applicable to each given non-terminal. This is done by removing and adding rules to the initial inherited grammar. Rules are numbered after changing the grammar and before each derivation step, in this way ensuring a deterministic genotype to phenotype mapping.

4. A solution to the p-median model using standard grammar evolution

In our first experiment, we used grammar evolution to solve the location-allocation problem by means of the p-median model. The grammar used was the following:

- Terminal symbols: set of all the allocation points = {1, 2, 3, …, N}
- Non-terminal symbols: {S, A1_1, … A1_(N-p), A2_1, … A2_(N-p), … A(p-1)_1, … A(p-1)(N-p)}
- Axiom: S
- The rules generate all the possible ordered sets of the p-medians, i.e. subsets of cardinal p of the set of terminal symbols in lower-to-higher order, without repetition.

The axiom (S) generates either median 1, followed by A1_1 (which generates p-1 medians from the set {2…N}), or median 2, followed by A1_2 (which generates p-1 medians from the set {3 … N}), … or median N-p, followed by A1_(N-p) (which generates p-1 medians from the set {N-p+1 … N}, or alternatively the set of p-medians N-p+1, N-p+2, … N.

The Ai_j rules are built in the same way as a function of the A(i-1)_j symbols, which generate one median less. Eventually, the derivations of this grammar generate all the possible correct p-median subsets.

We have performed 25 experiments of grammar evolution with this grammar, working on a set of 5000 demand points and a set of 40 allocations points, from which 20 should be selected (i.e. N=5000, n=40, p=20). Every experiment was run for 1500 generations. In every generation, only the best two individuals in the population of 100 tentative solutions made it through to the next generation. The other 98 were replaced by new offspring. The average best initial distance between the demand points and their allocated p-medians was always greater than 98.5 units. After the 1500 generations of grammar evolution, the best average distances came to be in the interval [95.18-96.53]. The evolution of the average distance as a function of the number of generations is very similar to that described in the next section and shown in figure 1.

5. A solution to the p-median model using Christensen grammar evolution

In our second experiment, we used Christensen grammar evolution to solve the location-allocation problem by means of the p-median model. The grammar used was the following:

$$
\begin{aligned}
G= \{\ & \{S(\downarrow g_i, \uparrow g_o),\ A(\downarrow g_i, \uparrow g_o, \downarrow m_i, \uparrow m_o)\ \},\\
& \{1, 2, 3, \ldots, 40\},\\
& S,\\
& P\}
\end{aligned}
$$

where P is made of the following rules:

$$
\begin{aligned}
S(\downarrow g_0, \uparrow g_{20}) \rightarrow\ & A_1(\downarrow g_0, \uparrow g_1, 21, \uparrow m_1)\\
& A_2(\downarrow g_1, \uparrow g_1, \downarrow m_1, \uparrow m_2)\\
& \ldots\\
& A_{19}(\downarrow g_{18}, \uparrow g_{19}, \downarrow m_{18}, \uparrow m_{19})\\
& A_{20}(\downarrow g_{19}, \uparrow g_{20}, \downarrow m_{19}, \uparrow m_{20})\}
\end{aligned}
$$

$$
\begin{aligned}
\{\ & A(\downarrow g_i, \uparrow g_o, \downarrow m_i, \uparrow m_o) \rightarrow i\\
& \{\ g_o = g_i\ - \{A \rightarrow j\}_{1 \le j \le i} \cup \{A \rightarrow m_i + 1\};\\
& m_o = m_i + 1\ \}\}_{1 \le i \le 21}
\end{aligned}
$$

This grammar generates the same language than the one of the previous section. The axiom S has two attributes: the initial grammar and the synthesized one. The non terminal A has four attributes distributed in two pairs, the first one takes into account the grammars

used in the derivations; the second one is devoted to the right hand side of the rules added after each derivation. The first attribute of each pair is inherited while the second one is synthesized. The first rule generates a word containing 20 copies of the non-terminal A. Its semantic actions propagate the grammars and the right hand sides of the new rules from left to right.

Each A can initially produce a number in the set {1,..,21} because the derivation process goes left to right and the first facility cannot be greater than 21. After applying any rule to a non-terminal A (producing, for example the facility m) all the rules giving a facility less or equal than m have to be removed. The derivation goes left to right, so the ith non-terminal A is derived after the i-1th A. Given that the right hand side of every new rule is propagated by means of a couple of attributes, the synthesized grammar of each A symbol must contain a new rule for A whose right hand side is computed from the attributes of the left hand side of the rule applied.

We have performed 25 experiments of Christensen grammar evolution with this grammar, working on a set of 5000 demand points and a set of 40 allocations points, from which 20 should be selected (i.e. N=5000, n=40, p=20). Every experiment was run for 1500 generations. In every generation, only the best two individuals in the population of 100 tentative solutions made it through to the next generation. The other 98 were replaced by new offspring. The average best initial distance between the demand points and their allocated p-medians was always greater than 98.5 units. After the 1500 generations of Christensen grammar evolution, the best average distances came to be in the interval [94.93-96.92]. Figure 1 shows the evolution of the average distance as a function of the number of generations in the experiment that generated the best result.

It can be observed that the average distance follows an approximate Poisson curve. In a previous work on the automatic generation of fractal curves with a given fractal dimension [Cebrian et al 2004], we proposed a procedure to make the genetic algorithm which we were using, in a grammar evolution context, increase its performance by about one order of magnitude. This procedure made use of the fact that the time needed to reach the goal in that case is not a normal distribution, but a heavy-tail one. Thus, a strategy based on stopping the algorithm and reinitializing it, when it has not reached an acceptable goal after a certain number of generations, gives rise to very good performance improvements. With this procedure in view, we have analyzed the situation for the case of the p-median location-allocation problem, but have come to the conclusion that, although it is possible that the distributions may still be heavy-tail, the performance improvement reached by applying the re-initialization procedure will be minimal, if any, because the minimum number of generations to reach an acceptable goal seems to be very large. This means that re-starting the algorithm does not provide us with a better chance of reaching an acceptable goal in a short time.

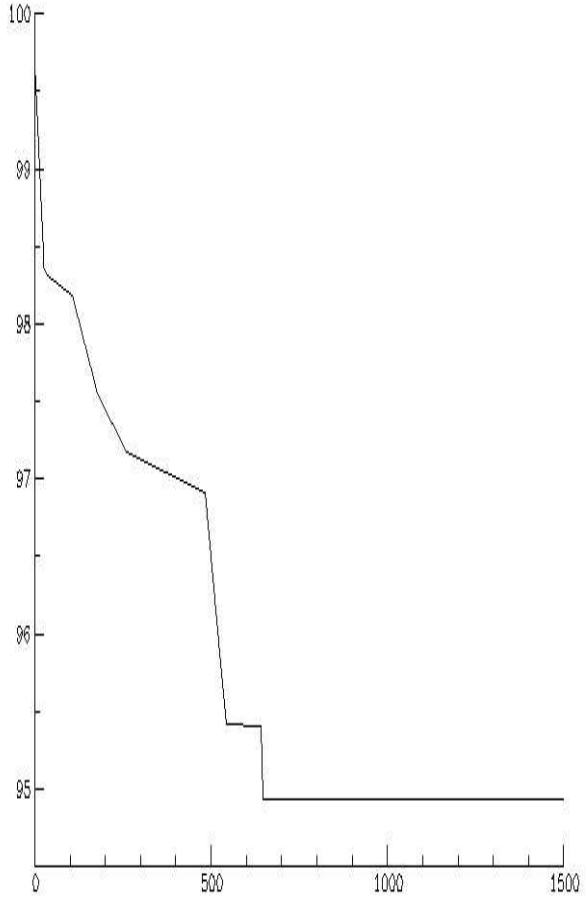

Figure 1. Average distance as a function of the number of generations in the Christensen grammar evolution solution to the p-median location-allocation problem.

6. Conclusions and future work

In this paper, we propose a new algorithm to solve the p-median problem. The algorithm uses grammar evolution or Christiansen grammar evolution to find a good solution to the problem. Christiansen grammar evolution starts from a much easier to design grammar, made of only 22 rules, while the standard grammar evolution version has to work with over 4500 rules. Both procedures reached comparable results: it can be mentioned that the best two solutions obtained were got with the Christiansen evolution grammar, but this procedure also led to the worst solution. From the point of view of performance, both alternatives came to be about comparable.

In the future, we intend to compare our two approaches to the use of standard genetic algorithms or the tabu algorithm, performed by other authors [Correa et al 2001], [Lorena and Lopes 1996]. We also will apply the procedure to real problems, such as finding the best distribution of locations for cellular telephone antennae, or for bank ATM in the city of Amman. In

fact, the trend towards using location-allocation modelling to support decision making for different organizations in the world, is becoming very important for planning purposes and in many leading countries is now at the implementation stage. The need for such modelling is even greater in developing countries, for instance, to determine vital GIS locations. This is the reason why we intend to implement our new algorithm to support decision making to determine ATM bank locations. This is important, since there are no existing planning strategies for such purposes with a low-cost price.

References

[Aho et al 1986] Aho, A.V., Sethi, R., Ullman, J.D. 1986: *Compilers: Principles, Techniques, and Tools*. Addison-Wesley, Reading, MA.

[Cebrián et al 2004] Cebrián, M., Ortega, A., Alfonseca, M. 2004: Acceleration of a procedure to generate fractal curves of a given dimension through the probabilistic analysis of execution time, in *Intelligent Engineering Systems Through Artificial Neural Networks*, Vol. 14, ed. C.H.Dagli, A.L.Buczak, D.L.Enke, M.J.Embrechts, O.Ersoy, pp. 265-270, ASME Press, New York, 2004.

[Christiansen 1990] Christiansen, H. 1990: A Survey of Adaptable Grammars, *ACM SIGPLAN Notices*, vol. 25, nº11. November 1990, pp. 35–44.

[Correa et al 2001] Correa E. S., Steiner M. T. A., Freitas A. A., Carnieri C. 2001: A Genetic Algorithm for the p-median Problem, *Proceedings of the Genetic and Evolutionary Computation Conference GECCO 2001*, San Francisco, California, Morgan Kaufmann Publishers.

[Daskin 1995] Daskin, M. 1995: *Network and Discrete Location: Models, Algorithms and Applications*, John Wiley, New York.

[Densham and Rushton 1992] Densham, P., Rushton, G. 1992: A More Efficient Heuristic for Solving Large P-Median Problems, *Papers in Regional Science* 71, pp. 307–329.

[Garcia-Lopez et al 2002] Garcia-Lopez, F., Melian-Batista, B., Moreno-Perez, J., Moreno-Vega, J. 2002: The Parallel Variable Neighborhood Search for the P-Median Problem, *Journal of Heuristics*, 8(3), pp. 375-388.

[Hansen et al 2001] Hansen, P., Mladenovic, N., Perez-Brito, D. 2001: Variable Neighborhood Decomposition Search, *Journal of Heuristics*, 7(3), pp. 335-350.

[Hilsman 1984] Hilsman, E. 1984: The P-Median Structure as a Unified Linear Model for Location Allocation Analysis, *Environment and Planning A*, Vol. 16, pp. 305-318.

[Hodgson 1978] Hodgson, M. 1978: Toward More Realistic Allocation in Location-allocation Models: An Interaction Approach, *Environment and Planning A*, 10, pp. 1273-85.

[Koza 1992] Koza, J.R. 1992: *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, MIT Press, Cambridge, Massachusetts. 1992

[Lorena and Lopes 1996] Lorena, L., Lopes,.L.S. 1996: Computational experiments with genetic algorithms applied to set covering problems, *Pesquisa Operacional*, 16:41-53.

[Megiddo and Supowit 1984] Megiddo, N., Supowit, J. 1984: On the Complexity of Some Common Geometric Location Problems, *SIAM J. Computing* 13, pp. 182-196.

[O'Neill and Conor 2003] O'Neill, M., Conor, R. 2003: *Grammatical Evolution, evolutionary automatic programming in an arbitrary language*, Kluwer Academic Publishers.

[Resende and Werneck 2003] Resende, M., Werneck, R. 2003: On the Implementation of a Swap-Based Local Search Procedure for the P-Median Problem, In R. E. Ladner, editor, *Proceedings of the Fifth Workshop on Algorithm Engineering and Experiments* (ALENEX'03), pp. 119-127. SIAM.

[ReVelle and Swain 1970] ReVelle, C.S., Swain, R.W. 1970: Central Facilities Location, *Geographical Analysis 2*, pp. 30–42.

[Shutt 1993] Shutt, J. N. 1993 : Recursive Adaptable Grammars. A thesis submitted to the Faculty of the Worcester Polytechnic Institute in partial fulfillment of the requirements for the degree of Master of Science in Computer Science, August 10 (amended December 16, 2003)

[Teitz and Bart 1968] Teitz, M., Bart, P. 1968: Heuristic Methods for Estimating the Generalized Vertex Median of a Weighted Graph, *Operations Research*, Vol. 16, 1968, pp. 955–961.

[Whitaker 1983] Whitaker, R. 1983: A Fast Algorithm for the Greedy Interchange of Large-Scale Clustering and Median Location Problems, *INFOR*, Vol. 21, pp. 95-108.