Article

A genetic algorithm approach to solving the anti-covering location problem

Sohail S. Chaudhry
Department of Decision and Information Technologies, Villanova School of Business, Villanova University, 800 Lancaster Avenue, Villanova, PA 19085, USA
E-mail: sohail.chaudhry@villanova.edu

Abstract: *In this paper we address the problem of locating a maximum weighted number of facilities such that no two are within a specified distance from each other. A natural process of evolution approach, more specifically a genetic algorithm, is proposed to solve this problem. It is shown that through the use of a commercially available spreadsheet-based genetic algorithm software package, the decision-maker with a fundamental knowledge of spreadsheets can easily set up and solve this optimization problem. Also, we report on our extensive computational experience using three different data sets.*

Keywords: facility location, anti-covering problem, genetic algorithm

1. Introduction

In this paper, we study a location problem known as the anti-covering location problem (ACLP). Given a finite number of potential facility location sites, the problem is to locate a maximum weighted set of facilities such that no two facilities are within a maximum distance or time of each other. The unweighted ACLP is equivalent to the maximum independent set problem in graph theory (Christofides, 1975; Balas & Yu, 1986). The purpose of this paper is to demonstrate that the methodology known as a genetic algorithm can be applied to solve the ACLP using a commercially available spreadsheet-based genetic algorithm software package. To the best of our knowledge, no research paper has been published in which a commercially available spreadsheet-based genetic algorithm software package is used to solve optimization problems. We test this approach on a randomly generated data set and two widely used data sets in the location literature that have been used to study a variety of different location problems, and our computational experience is reported.

The plan of the paper is as follows. In Section 2, the mathematical statement of the ACLP is presented. Section 3 provides a brief literature review of the location problem. A short discussion of a genetic algorithm is provided in Section 4. In Section 5, a generic solution approach is described for an optimization problem using a commercially available microcomputer-based genetic algorithm software package. The computational experience and discussion of the results are given in Section 6. The conclusions and future directions are presented in the last section.

2. The ACLP

Based on the ACLP formulated by Moon and Chaudhry (1984), two alternative integer programs were given by Murray and Church

(1997). For completeness, the mathematical formulation of the ACLP as presented by Murray and Church (1997), which differs slightly from the original formulation of the ACLP, is as follows:

maximize

$$Z = \sum_j p_j x_j$$

subject to

$$n_j x_j + \sum_{k \in Q_j} x_k \leq n_j \text{ for all } j \in M \qquad (1)$$

$$x_j = (0, 1) \text{ for all } j \in M \qquad (2)$$

where $x_j = 1$ if a facility is located at site j and 0 otherwise; $M = \{1, 2, \ldots, m\}$, the index set for potential facility sites; $Q_j \notin M$, the set of sites which are closer than a specified minimum distance b from site j, for all $j \in M$; p_j is the profit (weight) associated with site j, $p_j \geq 0$; and n_j is the minimum coefficient necessary to impose locational restrictions.

In the above formulation, the objective function represents the maximum weighted number of facility sites selected. Constraint (1) represents the desired minimum separation requirement in the sense that, if a site j $(x_j = 1)$ is selected, then no other site within the desired distance will be included in the solution; in other words, it forces $\sum_{k \in Q_j} x_k = 0$. Constraint (2) places the integrality requirements on the decision variables. By setting $n_j = N$, where N is a large positive number, this formulation reduces to that of Moon and Chaudhry (1984). In-depth discussions of this issue are presented in the work of Murray (1995) and Williams (1990).

3. Literature review

The ACLP is an important problem with numerous practical applications in a variety of fields: homeland security and military defence location (Moon & Chaudhry, 1984; Chaudhry *et al.*, 1986); telecommunications and computer vision (Balas & Yu, 1986); forest management (Barahona *et al.*, 1992); and DNA sequence matching (Joseph *et al.*, 1992).

In the facility location literature, some research has been reported on the ACLP. Moon and Chaudhry (1984) studied the ACLP in the context of locating facilities on a general network. An integer program for the ACLP was developed and the linear programming relaxation method was used to solve the problem. This approach parallels the approach taken by Toregas *et al.* (1971) for the set covering problem. Due to fractional results in the decision variables, a secondary constraint was then introduced into the formulation based upon the linear programming objective function value. However, the reported computational results were disappointing. Chaudhry *et al.* (1986) presented four heuristics and their worst case analysis for the ACLP, reporting extensive computational experience with relatively large size randomly generated problems of up to 50 potential facility sites. Also, their results for the small and medium sized problems were compared with the exact integer solutions. The analysis showed that some of the heuristics performed quite satisfactorily on these randomly generated problem sets.

Murray and Church (1997) used the Lagrangian relaxation approaches to solve the ACLP. Two different solution approaches were developed and tested on four different data sets. It was shown that the solution technique produced desirable results and in only two out of the 41 test problems did the technique fail to produce optimal solutions. In both the non-optimal cases, the deviation from the optimal solution was one unit.

More recently, Díaz-Báñez and Gómez (2000) studied the ACLP but they addressed the problem on a plane. Their general approach consisted of transforming the planar ACLP into the problem of finding the shortest path avoiding obstacles. Efficient algorithms were developed and applied to a variety of different problem situations.

4. Brief description of a genetic algorithm

A genetic algorithm is a heuristic search procedure which is based on the natural process of evolution as in biological sciences and was first introduced by Holland (1975). As this highly adaptive evolutionary process progresses, the population genetics evolve in a given environment according to the natural behavior in which the fittest survive and the weakest are destroyed. Thus, the genes from the adept donor will then propagate to other recipients during each successive generation, creating more apt offspring suitable for the defined environment. In optimization terms, the search algorithm improves the solution over generations and as it progresses towards the optimum. Genetic algorithms have been successfully applied in solving a variety of optimization problems which are difficult to solve, including the traveling salesperson problem, job-shop scheduling problems, vehicle routing problems, airline crew scheduling problems, optimizing the sequence of advertisements within a commercial break at a British television station, and painting trucks at a General Motors production facility, among others (Chaudhry *et al*., 2000). For a more thorough coverage of genetic algorithms, the reader is referred to the excellent textbook by Goldberg (1989).

In terms of an optimization problem, the genetic algorithm approach is summarized as follows. At any given point in time, the genetic algorithm generates a population of possible candidate solutions. Initially, the population size is chosen at random. However, this choice typically depends on the characteristics of the problem. Each population component is a string entity of chromosomes, which represents a possible solution to the problem. The population components are evaluated based on a given objective function. Highly fit population components are given the chance to reproduce through a crossover process with other highly fit population elements by exchanging pieces of their genetic information. This process produces 'offspring' or new solutions to the optimization problem based upon the high performance characteristics of the parents. A mutation process prevents premature loss of important information by randomly altering bits within a chromosome. This procedure continues until a satisfactory solution is achieved.

5. A generic solution approach

To the best of our knowledge, there are three commercially available software packages, namely Evolver (1997), GeneHunter (1995) and Generator (1995), that are based on the basic principles of a genetic algorithm as described above. There are two ways that could be used to implement optimization problems, namely using an Excel spreadsheet add-in or programming one's own genetic algorithm in either Visual Basic or C language using the built-in Dynamic Link Library functions. Using the ACLP, the first approach was utilized and an attempt was made to see if a manager with spreadsheet knowledge could set up optimization problems and use the genetic algorithm software package to obtain good solutions in a reasonable amount of time.

Once the optimization problem was modeled into an Excel spreadsheet and the software package was linked using the add-in feature in Excel, the problem-solving parameters were then specified in a dialog box. The first item in the dialog box was the fitness function cell. The value in this cell contains the formula that measures the success in finding a solution to the problem. The success is measured by choosing the maximum, the minimum or a selected value for the adjustable chromosomes, which eventually produced the most favorable number in the fitness function cell. Chromosomes are the variables whose values are adjusted in order to solve the optimization problem. The software packages allow for chromosomes to be either continuous or enumerated. Also, the dialog box allows the user to enter either hard or soft constraints.

In the next step, the software packages allow the user to interface with parameters that drive the evolution process. Initially, the software Excel interface sets the various parameters to default settings, which are quite appropriate for many applications. However, a user can change these parameters as needed for experimentation. Under population, the software packages allow the user to change the population size, which represents the number of individuals in the population. Chromosome length is the number of genes in a chromosome. Larger length implies higher precision answers; however, it takes longer to get the possible solutions. The evolution parameters are the crossover rate, usually set to a high value of probability, the mutation rate, which is usually set to a low value of probability, and the generation gap, typically set at a high value between 0 and 1 so that only a selected few individuals go from one generation to the next. The software packages allow the user to choose various types of strategies when the generation is set, such as elitism and diversity. The other choices are screen update and when to stop the evolution process.

Once the various parameters are set, the software packages allow the user to save, restore and delete the model as well as to reset all options to default values. To solve the problem, the user clicks on the start button and the solution process begins. The software packages provide the user with the flexibility of stopping the process and changing parameters. For example, one could use the strategy of starting with a chromosome length of 8 bits until little progress is made towards a better solution, and then interrupting the evolutionary process and changing it to a 16-bit chromosome and continuing the search. Another example would be to start with the elitism strategy off to allow the population to evolve without significant selective pressure. After some time, the elitism strategy can be turned on such that the evolutionary process evaluates around the best solutions.

6. Computational experience and discussion

Chaudhry (1998) tested the three commercially available spreadsheet-based genetic algorithm software packages, GeneHunter, Evolver and Generator, on a variety of different optimization problems and found that the GeneHunter and Generator software packages performed better, in terms of the solution value and computational time, than Evolver. Thus, GeneHunter was used to test the ACLP using three different data sets.

The ACLP problem was analyzed utilizing GeneHunter with a randomly generated data set and also with two frequently used data sets in the location literature. The first data set used was the randomly generated data set, a 20-node set, from the study of Chaudhry *et al.* (1986). The second was a widely used data set based on distances between largest cities in New York State, a 30-node set, from Toregas *et al.* (1971). The third data set was the Washington, DC, 55-node set from Swain (1971). All computational analysis was conducted using a Pentium processor based PC under the Windows operating environment. It should be noted that in such studies a problem instance is typically solved multiple times and, depending upon the problem size, the computational burden could be as large as an hour for a problem instance! In this study, the computational experience presented is based on 41 problems with a total of 410 problem instances using the three data sets. However, we also tested these problems under different sets of parameters with no appreciable improvement in the results. The results from our experimentation of using genetic algorithms to solve the ACLP were then compared with those of Murray and Church (1997) in which optimal solutions were identified for the New York State and Washington, DC, data sets; the results based on the randomly generated data set were compared to the study of Chaudhry *et al.* (1986).

Table 1 summarizes the results based on the randomly generated data set consisting of 20 potential facility sites. A total of 20 ACLPs were solved by varying the minimum separation distance b between 50 and 1000 in increments of 50. The optimal solution for these problems was available from Chaudhry *et al.* (1986). Each problem instance was solved ten times using the default parameters of GeneHunter except that the population size was set at 50 and the solution was set to terminate after 1000 generations. As indicated in Table 1, the optimum results were obtained for all the solved problem instances. The computational burden was relatively small and the convergence took place in a reasonable number of generations as indicated in Table 1.

The results from the second data set of Toregas *et al.* (1971) with 30 potential facility sites are shown in Table 2. A total of seven ACLP

Table 1: *Results for the 20 × 20 randomly generated data set*

Minimum distance between facilities (b)	*Best genetic algorithm solution out of ten runs using GeneHunter*	*Optimal solution from Chaudhry et al. (1986)*	*Average CPU time of ten runs (s)*	*Average number of generations based on ten runs*
50	12	12	111	44
100	12	12	125	46
150	9	9	136	47
200	9	9	148	51
250	8	8	139	49
300	7	7	119	47
350	6	6	120	42
400	6	6	115	45
450	6	6	76	49
500	5	5	77	48
550	5	5	72	46
600	5	5	85	54
650	5	5	88	55
700	3	3	68	43
750	3	3	69	44
800	3	3	75	47
850	3	3	76	49
900	3	3	72	45
950	2	2	63	41
1000	1	1	68	41

problems were solved by setting the minimum separation distance *b* at 35, 50, 100, 150, 200, 300 and 400. The optimal solution for all the problem instances was available from Murray and Church (1997). Once again, each problem instance was solved ten times using the default parameters of GeneHunter except that the population size was set at 50 and the solution was set to terminate after 1000 generations. As indicated in Table 2, the optimum solution was obtained for all the solved instances. However, it should be noted that, for the case when the minimum separation *b* was set at 100, the optimal solution was not obtained at the population size setting of 50 but at a population size setting of 100. This effect can be observed by the higher computational burden associated with this problem instance. Again, except for one case, the computational burden was relatively small and it took no more than 67 generations to obtain the optimal solution.

For the 55 potential facility sites data set of Swain (1971), 14 problem instances were solved using GeneHunter for which the optimal solutions were available from Murray and Church (1997). The results associated with this computational experience are shown in Table 3. These ACLPs were solved by setting the minimum separation distance *b* at 5, 6, 7, 8, 9, 10, 12, 14, 15, 20, 23, 27, 31 and 34. Once again, each problem instance was solved ten times using the default parameters of GeneHunter. However, each instance was solved five times with the population size parameter set at 50 and the remaining five with the population size parameter set at 100. This was done for this data set because GeneHunter was only able to achieve optimality for five of the 14 problem instances. For the remaining nine problem instances, the difference from the optimal solution was by one facility. Again, we made an attempt to resolve the non-optimality issue by changing the input parameters of GeneHunter but the results did not achieve the desired optimal solutions. The execution time and number of generations per problem instance on average were the highest

Table 2: *Results for the 30 × 30 New York State data set*

Minimum distance between facilities (b)	*Best genetic algorithm solution out of ten runs using GeneHunter*	*Optimal solution from Murray and Church (1997)*	*Average CPU time of ten runs (s)*	*Average number of generations based on ten runs*
35	24	24	214	57
50	21	21	229	62
100	10	10	508	67
150	6	6	227	56
200	4	4	211	51
300	3	3	205	55
400	2	2	201	53

Table 3: *Results for the 55 × 55 Washington, DC, data set*

Minimum distance between facilities (b)	*Best genetic algorithm solution out of ten runs using GeneHunter*	*Optimal solution from Murray and Church (1997)*	*Average CPU time of ten runs (s)*	*Average number of generations based on ten runs*
5	36	36	1029	96
6	30	30	937	89
7	24[a]	25	1039	99
8	20[a]	21	1044	99
9	17[a]	18	987	111
10	15[a]	16	874	112
12	12[a]	13	748	98
14	10[a]	11	872	111
15	9	9	971	89
20	6[a]	7	710	88
23	5[a]	6	687	89
27	4[a]	5	656	85
31	4	4	823	78
34	3	3	780	71

[a]Non-optimal solution using GeneHunter with a difference of one facility.

among the three data sets analyzed in this computational experience.

Hence, in summary, of the 41 ACLPs analyzed in our computational experience, the genetic algorithm of GeneHunter was able to find the optimal solution for 32 problem instances. For the remaining 9 problem instances for which the spreadsheet-based genetic algorithm failed to determine the optimal solution, the deviation from the optimal solution was only off by a single facility. This is an interesting observation since the same deviation was reported by Murray and Church (1997) where the Lagrangian relaxation methodology was used to solve the ACLP. In addition, the execution times for all the problems of the three data sets ranged from nearly one minute to less than 18 minutes with the average number of generations as low as 41 to as high as 112.

7. Conclusion

In this paper we solved the anti-covering facility location problem using GeneHunter, a commercially available spreadsheet-based genetic algorithm software package. Our computational experience, using three different data sets, shows that of all the problems tested around 78% of the time an optimal solution was obtained in a reasonable execution time. For the remaining problems, the solution generated by the software was no worse than one facility site. In order to address the issue of non-optimality for almost 22% of the problems, it seems quite appropriate that a dedicated genetic algorithm must be developed to specifically address the uniqueness of the anti-covering facility location problem. It has been shown by Michalewicz (1994) that it is important to design genetic algorithms based on the problem specified and this will be addressed in future research.

Acknowledgement

This research was partially supported by a grant from the National Science Foundation (0332490).

References

BALAS, E. and C. YU (1986) Finding a maximum clique in an arbitrary graph, *SIAM Journal on Computing*, **15**, 1054–1068.

BARAHONA, F., A. WEINTRAUB and R. EPSTEIN (1992) Habitat dispersion in forest planning and the stable set problem, *Operations Research*, **40**, S14–S21.

CHAUDHRY, S.S. (1998) An evaluation of genetic algorithm software packages, presented at the CORS/INFORMS National Meeting at Montreal, 26–29 April.

CHAUDHRY, S.S., S.T. MCCORMICK and I.D. MOON (1986) Locating independent facilities with maximum weight: greedy heuristics, *OMEGA*, **14**, 383–389.

CHAUDHRY, S.S., M.W. VARANO and L. XU (2000) Systems research, genetic algorithms, and information systems, *Systems Research and Behavior Science*, **17**, 149–162.

CHRISTOFIDES, N. (1975) *Graph Theory: An Algorithmic Approach*, New York: Academic Press.

DÍAZ-BÁÑEZ, J.M. and F. GÓMEZ (2000) Anti-covering shortest path problem in the plane, presented at the EURO Working Group on Locational Analysis – EWGLA XII Meeting, Barcelona, 14–17 December.

EVOLVER: ADVANCED GENETIC OPTIMIZATION for SPREADSHEETS (1997) Palisade Corporation, Newfield, New York, NY.

GENEHUNTER (1995) Ward Systems Group Inc., Frederick, MD.

GENERATOR (1995) New Light Industries Ltd, Spokane, WA.

GOLDBERG, D.E. (1989) *Genetic Algorithms in Search, Optimization, and Machine Learning*, Reading, MA: Addison-Wesley.

HOLLAND, J. (1975) *Adaptation in Natural and Artificial Systems*, Ann Arbor, MI: University of Michigan Press.

JOSEPH, D., J. MEIDANIS and P. TIWARI (1992) Determining DNA sequence similarity using maximal independent set algorithms for interval graphs, in *Algorithm Theory – SWAT '92*, O. Nurmi and E. Ukkonen (eds), Berlin: Springer.

MICHALEWICZ, Z. (1994) *Genetic Algorithms + Data Structures = Evolution Programs*, 2nd edn, Berlin: Springer.

MOON, I.D. and S.S. CHAUDHRY (1984) An analysis of network location problems with distance constraints, *Management Science*, **30**, 290–307.

MURRAY, A.T. (1995) Modeling adjacency conditions in spatial optimization problems, PhD Dissertation, Department of Geography, University of California at Santa Barbara.

MURRAY, A.T. and R.L. CHURCH (1997) Solving the anti-covering problem using Lagrangian relaxation, *Computers and Operations Research*, **24**, 127–140.

SWAIN, R.A. (1971) A decomposition algorithm for a class of facility location problems, PhD Dissertation, Cornell University, Ithaca, NY.

TOREGAS, C., R.W. SWAIN, C.S. REVELLE and L. BERGMAN (1971) The location of emergency service facilities, *Operations Research*, **19**, 1363–1373.

WILLIAMS, H. (1990) *Model Building in Mathematical Programming*, 3rd edn, Chichester: Wiley.

The author

Sohail S. Chaudhry

Sohail S. Chaudhry received his PhD in industrial engineering and operations research from Columbia University. His research interests include network facility location, management and control of quality, and vendor selection. His research has appeared in numerous journals including *Decision Sciences*, *European Journal of Operational Research*, *Expert Systems*, *International Journal of Production Research*, *Journal of the Operational Research Society*, *International Journal of Quality and Reliability Management*, *Management Science*, *Omega* and *Systems Research and Behavioral Science*.