Journal of the Arkansas Academy of Science

Volume 49 Article 23

1995

Construction Resource Allocation Using a Genetic Algorithm

Siripong Malasri
Christian Brothers University

Jennifer R. Martin
Christian Brothers University

Construction Resource Allocation Using a Genetic Algorithm

Siripong Malasri and Jennifer R. Martin
Department of Civil Engineering
Christian Brothers University
Memphis, TN 38104

Abstract

A proper allocation of limited resources (men, machines, materials, and money) is critical in a construction project. Traditionally, resource allocation problems have been solved using methods in operations research (OR), such as mathematical programming. In recent years, genetic algorithms (GA) have emerged as an effective optimization methodology. One major advantage of the GA approach over the OR approach is that the GA approach is universal for various types of optimization problems, unlike the OR approach which varies depending on the types of problems at hand. This paper shows an application of GA to a resource allocation problem in the construction industry in which a contractor tries to maximize profit by properly allocating various pieces of heavy equipment to various ongoing construction projects. This type of problem has customarily been solved by the linear programming method. GA has proved to be quite an attractive alternate to the OR method. Since the GA method is more universal than the OR method, the program can be easily modified to solve other types of problems. A description of a computer program written in Visual Basic is also presented.

Introduction

Resource allocation has been a challenging problem for many organizations. At present the construction industry is highly competitive. A proper allocation of limited resources (men, machines, materials, and money) is critical to their success. The profit margin has been shrinking and heavy equipment is expensive. Therefore, due to the multi-project nature of the industry, it is preferable to properly manage fewer pieces of equipment than it is to acquire more pieces of equipment.

Traditionally, resource allocation problems have been solved using methods in operations research (OR), such as mathematical programming (Hillier and Lieberman, 1974). In recent years, genetic algorithms (GA) have emerged as an effective optimization methodology (Michalewicz, 1992). One major advantage of the GA approach over the OR approach is that the GA approach is universal for various types of optimization problems, unlike the OR approach which varies depending on the types of problems at hand. This paper shows an application of GA to a resource allocation problem in the construction industry in which a contractor tries to maximize the profit by properly allocating various pieces of heavy equipment to various ongoing construction projects.

Materials and Methods

Genetic algorithms (GA) are based on the natural selection process. The process starts with a randomly created first generation of population. The population is usually kept at a constant size for the entire evolution to simplify the process. Every individual in a generation represents one solution. An individual consists of one chromosome with a number of genes. Each chromosome is then evaluated for its fitness – how well it accomplishes the set goal. More fit chromosomes have a better chance to get into the next generation. Genes are exchanged through the crossover process and diversity is added into the population by the mutation process. The process is repeated over several generations and the overall best solution is used. The general procedure is very similar to previous work (Malasri et al., 1994) with differences in the details of each step. The following sections describe the details of this process in the context of resource allocation optimization.

Problem Statement.––Generally, the objective of the resource allocation problem is to maximize a function P of n variables under m constraints:

$$P = P(X_1,X_2,X_3,...,X_n)$$

under the following constraints:

$$f_1(X_1,X_2,X_3,...,X_n)\leq 0$$
$$f_2(X_1,X_2,X_3,...,X_n)\leq 0$$
$$f_3(X_1,X_2,X_3,...,X_n)\leq 0$$
$$...$$
$$f_m(X_1,X_2,X_3,...,X_n)\leq 0$$

As an example, a problem previously solved by linear programming (Pilcher, 1976) is used: "A contractor has one mechanical excavator and one bulldozer which are available for work on either of two adjacent sites. On one site clay overburden is being excavated for a ballast pit owner and on the other, ballast is being removed under

subcontract to another client. The contractor's experience leads him to believe that he can make £50 profit for every 1000 m^3 of clay overburden and £60 profit for every 1000 m^3 of ballast he removes. A comprehensive work study assesses the resources required to remove 1000 m^3 of clay to be 8 hours' use of the excavator, 4 hours' use of the bulldozer and 50 man-hours of laborers' time. In the case of the excavation of 1000 m^3 of ballast, the resources are required for 4 hours, 5 hours and 13 man-hours respectively. The contractor's employees work a 40-hour week. The mechanical equipment is also available for a 40-hour week. In addition to the mechanical equipment, 5 laborers are available for up to 40 hours each in any one week in order to assist with the work. When not employed on the excavation, use can be made of the laborers elsewhere. Question: how should the contractor use his resources in order to maximize his profit during one working week?"

If X_1 and X_2 represent the units of 1000 m^3 of clay and ballast excavated, respectively (both are positive value), the problem can be rewritten as:

$$\text{Maximize } P(X_1,X_2,) = 50\,X_1 + 60\,X_2$$

subject to the following constraints:

$$f_1 = 8\,X_1 + 4\,X_2 - 40 \leq 0$$
$$f_2 = 4\,X_1 + 5\,X_2 - 40 \leq 0$$
$$f_3 = 50\,X_1 + 13\,X_2 - 200 \leq 0$$

Chromosome Formulation.--The first part of a chromosome contains the binary representation of the first variable (X_1) while the second part represents the second variable (X_2). Let a_i and b_i be the lower and upper bound of the i-th variable. The range of the i-th variable (r_i) becomes (b_i-a_i). If d is the number of decimal places on each variable, then the number of genes required for the binary representation of the i-th variable (g_i) is:

$$g_i = \text{In}\,(r_i * 10^d) / \ln(2)$$

The derivation of the above equation for g_i can be found in a textbook example (Michalewicz, 1992). The chromosome consists of a total of ($g_1 + g_2$) genes, since there are only two variables (X_1 and X_2).

The First Generation.--The first generation is randomly created by filling all gene slots with 0 or 1. For each gene slot, a random number (between zero and one) is generated. If this number is less than 0.5, the value 0 is entered into the gene slot, otherwise the value 1 is entered. To ensure that the random number does not always start from the same point in the random number sequence, the program uses the minute part of the computer clock to seed the starting location in the random number sequence.

Fitness Evaluation.--Each chromosome is evaluated for its fitness in order to determine the chance of being selected into the next generation. From each chromosome, the binary representation (base 2) of each variable is converted into a corresponding real number (base 10). These real numbers, after divided by 10^d, are then entered into the objective function to determine the value of the function P. Each constraint condition is then evaluated. Initially, the fitness is set equal to the value of function P. Thus, the higher the value of the function P, the higher the fitness. For each violation of constraint conditions (i.e., the value of f_i is greater than zero), the fitness is reduced by dividing the fitness by the user-specified extra penalty. It also could occur that the value of X_i turns out to be greater than the user-specified upper limit. If this happens, the fitness for that chromosome is also reduced by dividing the fitness by the same extra penalty.

Population Selection, Cross Over, and Mutation.--Chromosomes are selected into the next generation based on their fitness. The process is similar to creating a spinner in which a chromosome with a larger fitness occupies a larger area on the spinner. When the spinner is spun, it would have a greater chance to stop on a larger area. The spinner is then spun for the number of population to select a group of potential parents. These potential parents then go through the process of crossing over (in which genes are exchanged) and the process of mutation (in which genes are altered). The details of these procedures can be found in a recent work (Malasri et al., 1994).

Results and Discussion

The method described above was implemented in a computer program using Microsoft Visual Basic programming language which operates under the Windows 3.1 environment. Figure 1 shows the evolution screen. The left and upper parts of the screen are the user input area for both problem-specific input and genetic parameters. In the middle of the screen are the results of the evolution. The program displays the best solution for the current generation as well as the overall best solution from the first generation to the current generation. A graph is also displayed to show the distribution of all solutions in each generation where each dot represents a possible solution. In the first generation, the possible solutions (chromosomes) are created randomly. Thus the dots are spread over the entire graph area. As generations pass, these dots are concentrated only in a few areas of the graph which shows that most solutions converge to the location of the maximum function.

The program was executed for 40 runs consecutively with varied parameter values. For comparison, the best

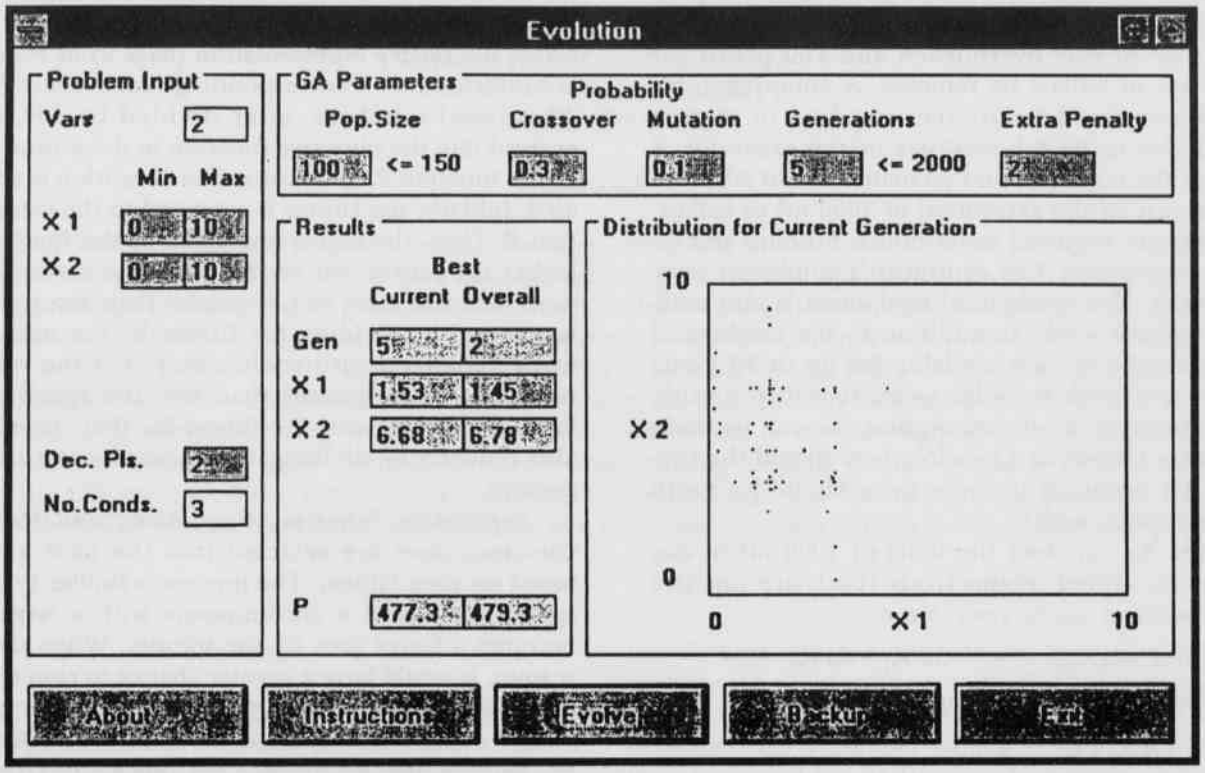

Fig. 1. Evolution screen.

Table 1. Results from 40 consecutive runs.

Crossover Probability = 0.3 Mutatio Probability = 0.1

Run No.	X_{1max}	X_{2max}	Dec. Pls.	Pop. Size	No. of Gen.	Ext. Penalty	Results Gen.	Results X_1	Results X_2	Results P	% Error*
1	10	10	2	10	50	2	42	2.40	4.56	393.6	-18.56
2	10	10	2	10	50	2	44	2.31	4.98	414.3	-14.28
3	10	10	2	25	50	2	42	2.39	5.04	421.9	-12.71
4	10	10	2	25	50	2	13	0.79	7.31	478.1	-1.08
5	10	10	2	50	50	2	36	0.88	7.29	481.4	-0.40
6	10	10	2	50	50	2	24	0.78	7.37	481.2	-0.44
7	10	10	2	100	50	2	43	1.51	6.78	482.3	-0.21
8	10	10	2	100	50	2	4	1.57	6.73	482.3	-0.21
9	10	10	2	50	100	2	3	1.02	7.14	479.4	-0.81
10	10	10	2	50	100	2	59	2.44	5.11	428.6	-11.32
11	10	10	2	50	200	2	175	0.44	7.56	475.6	-1.60
12	10	10	2	50	200	2	32	1.45	6.81	481.1	-0.46
13	10	10	2	100	100	2	94	2.58	4.83	418.8	-13.35
14	10	10	2	100	100	2	50	1.61	6.71	483.1	-0.05
15	10	10	2	100	200	2	101	0.58	7.50	479.0	-0.90
16	10	10	2	100	200	2	7	1.12	7.09	481.4	-0.40
17	20	20	2	50	100	2	65	1.48	6.75	479.0	-0.90
18	20	20	2	50	100	2	16	1.06	7.15	482.0	-0.28

19	30	30	2	50	100	2	31	2.73	4.53	408.3	-15.52
20	30	30	2	50	100	2	56	2.47	5.06	427.1	-11.63
21	20	20	2	100	100	2	4	1.25	6.99	481.9	-0.30
22	20	20	2	100	100	2	2	1.58	6.50	469.0	-2.96
23	30	30	2	100	100	2	82	3.85	0.56	226.1	-53.22
24	30	30	2	100	100	2	28	2.99	3.83	379.3	-21.52
25	10	10	1	100	100	2	2	1.7	6.6	481	-0.48
26	10	10	1	100	100	2	8	1.0	7.2	482	-0.28
27	10	10	2	100	100	2	51	1.25	7	482.5	-0.17
28	10	10	2	100	100	2	44	1.12	7.09	481.4	-0.40
29	10	10	3	100	100	2	8	1.667	6.659	482.89	-0.09
30	10	10	3	100	100	2	84	2.938	4.077	391.52	-19.00
31	10	10	4	100	100	2	12	1.1037	7.09	480.58	-0.57
32	10	10	4	100	100	2	5	1.2228	7.0033	481.33	-0.41
33	10	10	2	100	100	1.5	10	0.74	7.32	476.2	-1.48
34	10	10	2	100	100	1.5	44	1.55	6.75	482.5	-0.17
35	10	10	2	100	100	2	26	1.21	7.03	482.3	-0.21
36	10	10	2	100	100	2	3	1.36	6.91	482.6	-0.15
37	10	10	2	100	100	2.5	19	1.53	6.77	482.7	-0.13
38	10	10	2	100	100	2.5	10	1.25	7	482.5	-0.17
39	10	10	2	100	100	3	91	0.6	7.52	481.2	-0.44
40	10	10	2	100	100	3	27	1.62	6.7	483	-0.07

*Best solution from linear programming method is 483.33

Table 2. Evolution from an initial random solution to the final solution.

Input Data:	Crossover Probability	=	0.3
	Mutation Probability	=	0.1
	X_{1max}	=	10
	X_{2max}	=	10
	Decimal Places	=	2
	Population Size	=	100
	Number of Generations	=	100
	Extra Penalty	=	2

Evolution Progress:

Generation	X1	X2	Profit
1	1.40	6.36	451.6
2	1.56	6.36	459.6
3	0.69	7.26	470.1
7	1.40	6.86	481.6
48	1.56	6.75	483.0

solution from linear programming method results in a profit of £483.33. The results from the genetic algorithm program are summarized in Table. 1. Sixty-five percent of the total runs yield the results within 1% error. The percentage increases to 75% and 95% for error within 5% and 20% respectively. The best profit, resulting from run number 14, is 483.1 as compared to 483.33. All runs took less than two minutes (with most within one minute) on a 486 DX2-66 computer. Runs with larger populations size tend to converge very early. A larger population size means a larger search space, which increases the number of candidate solutions. However, earlier convergence is not necessarily better, since computation time increases with population size. An extra run was made and the evolution results were recorded and summarized in Table 2. In this particular run, the solution started with the 1st generation's best solution of 451.6, evolved, and converged to a solution of 483 in the 48th generation.

Conclusions

Genetic algorithms work well with the resource allocation problems as shown in this paper. The major advantage of the GA method over the traditional OR methods is its applicability to a wide range of problems with very few modifications to the computer program. For example, the same computer program can be easily modified to solve non-linear programming problems or to solve problems with more than two variables. The GA method, however, does not guarantee the optimum solution. It is the user's responsibility to make several runs with different parameter values and then choose the best solution.

Literature Cited

Hillier, F.S. and **G.J. Lieberman,** 1974. Introduction to Operations Research. Holden-Day, Inc., San Francisco, 800 pp.

Malasri, S., D.A. Halijan, and **M.L. Keough,** 1994. Concrete Beam Design Optimization with Genetic Algorithms. Proc. Arkansas Acad. Sci.48:111-115.

Michalewicz, Z. 1992. Genetic Algorithms + Data Structures = Evolutionary Programs. Spring-Verlag, New York, 259 pp.

Pilcher, R. 1976. Principles of construction management. 2nd Ed., McGraw-Hill, Berkshire, 370 pp.