

A Minimal Bidding Application Solved By A Genetic Algorithm Where Element Costs Are Time Dependent

Joseph DeCicco
Pace University

Advisor: Michael L. Gargano; Committee Members: William Edelson and Allen Stix

1. Abstract

Researchers Gargano and Edelson (1) have developed five theoretical models to study the use of Genetic Algorithms (GA's) on optimal base models whereby the element costs are not fixed, but are time dependent. Here we consider the minimal Bidding problem. With the use of a Java program, we implement this problem using a GA to solve it. In addition, we focus on three objectives. First, we measure the sensitivity of Genetic Algorithms depending on variations in cost table. Next, we measure the sensitivity of Genetic Algorithms based on the size of the encoding. Finally, we measure the sensitivity of Genetic Algorithms depending on variations in population size verses generation size (size) and mutation rate.

2. Introduction

Traditionally, solving a minimal bidding problem involved determining the optimal solution by comparing the fixed value of each bid. Researchers' Gargano and Edelson (1) have considered an extension to having each bid not fixed, but rather time dependent. The cost of each bid depends on the order (i.e. the time) at which the bid is placed.

This model utilizes a genetic algorithm (GA). The genetic algorithm paradigm is an adaptive methodology based on Darwinian natural selection and genetic inheritance. It applies operations of selection (based on survival of the fittest), crossover (i.e. mating), and mutation to a given population of potential solutions to generate a new, more fit, population of potential solutions. After a number of generations, the process converges to an optimal or near optimal solution. The methodology is a very useful and powerful tool for efficiently generating answers to difficult problems (2).

This study will focus on three objectives in order to analyze the performance of genetic algorithms in solving this problem. First, the measure of the sensitivity of genetic algorithms depending on variations in the cost table is examined. Next, the sensitivity of genetic algorithms based on the size of the encoding is examined. Finally, variations in population size verses Generation size (size) and mutation rate is examined. All studies will involve a benchmark comparison of genetic algorithms against the brute force method.

3. Description of the Problem

A company is to construct a transportation network connecting a group of cities. Due to budget limitations only one link will be contracted for construction each month. Different contractors having different price scales, which vary over time, bid on the construction of the different links of the potential network. The cities together with all the possible links can be modeled to form a multigraph H . The goal is to construct the desired network (an underlying subgraph G of the multigraph H with q edges) such that the total cost over time of the entire construction is minimized. We also assume that it takes one unit of time to construct a link and that the network must be completed in the $t = q + s$ time periods available (q periods represent actual construction and s represents slack periods where no construction is performed). Dummy (slack) bids with zero cost for each period can be included. In this problem, we are searching for an underlying graph G whose q edges are sequenced over $q + s$ periods and whose total cost is minimum.

4. Mathematical Model

Given an undirected multigraph $H = (V, F)$ with no loops whose underlying simple graph $G = (V, E)$ has q edges, F consists of (m_i) multiple edges for each edge e_i in G . Each of $(m_1 + m_2 + \dots + m_q)$ edges in H can be thought of as a bid to build edge e_i in G . Let $E =$ all the edges in H and let the bases be $\beta = \{B \subseteq H \mid B = \{b_1, b_2, \dots, b_q\}$ contains exactly one bid for each edge e_i of $G\}$. Since the network must be completed in $t = q + s$ time periods, s ($s \geq 0$) dummy bids with zero cost for each period can be included to be used as slack periods where no construction is done (e.g., during winter when costs are prohibitive high). Thus we can expand the definition of β to a matroid of rank $q + s$ which includes these dummy contractor bids. Therefore, we now let the bases be $\beta = \{B \subseteq H \cup D \mid \text{where } B = \{b_1, b_2, \dots, b_{q+s}\}$ contain exactly one bid for each edge of G and also includes dummy (slack) bids $\}$ (1).

5. Genetic Algorithms and Fitness

Let us take an example from the description of the problem above. The figure below shows a transportation network consisting of four cities.

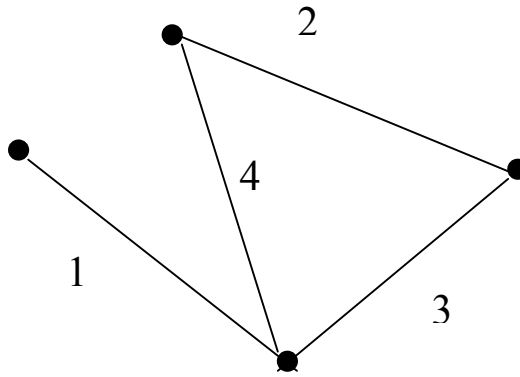


Figure 1

Now let us assume we have several bids for each of the 4 links to the cities. The table below shows the number of bids for links connecting each of the four cities.

| Link number | # of Bids obtained to connect the 2 cities | Bid numbers for all bids obtained to connect that link |
|-------------|--|--|
| 1 | 2 | 1 and 2 |
| 2 | 4 | 3,4,5,6 |
| 3 | 3 | 7,8,9 |
| 4 | 1 | 10 |

Table 1

Notice that there are four time periods because only one link will be built per time period. Also, there are ten total bids and several links have multiple bids. The table below shows the cost associated with each of the ten bids. It is the cost table used for the genetic algorithm.

| Bid # | T=1 | T=2 | T=3 | T=4 |
|-------|-----|-----|-----|-----|
| 1 | 83 | 59 | 95 | 33 |
| 2 | 69 | 87 | 32 | 20 |
| 3 | 55 | 85 | 39 | 74 |
| 4 | 29 | 87 | 73 | 34 |
| 5 | 92 | 36 | 43 | 83 |
| 6 | 51 | 59 | 62 | 35 |
| 7 | 22 | 21 | 13 | 96 |
| 8 | 87 | 43 | 43 | 36 |
| 9 | 69 | 87 | 55 | 43 |
| 10 | 82 | 74 | 18 | 24 |

Table 2

We will call the number of links S and the number of bids n. In the example above S is 4. We will need an array of numbers that has 8 positions to determine one possible solution. The first four positions determine the bids to be used and the second four positions determine the order to build them. As an example we will use the array (2,3,9,10,2,2,1,1) to encode one possible solution. To do this we take the permutation of the set of numbers. This is done by taking each position from left to right and using the number in the first four positions to determine bid numbers and the corresponding positions to determine the order.

In the example above (2,3,9,10,2,2,1,1), the first four values of the array (2,3,9,10) are used to determine which bid are used and the second four values of the array (2,2,1,1) are used to determine the order to build. So the permutation for the array (2,3,9,10,2,2,1,1) is (9,2,3,10). Figure 2 below illustrates how this is done.

| Position | Array | Encoding | |
|----------|--------------------|------------|---|
| 1 | (2,x,x,x,2,x,x,x) | (_,2,_,_) | Put number 2 in the 2nd open position from the left. |
| 2 | (x,3,x,x,x,2,x,x) | (_,2,3,_) | Put number 3 in the 2nd open position from the left. |
| 3 | (x,x,9,x,x,x,1,x) | (9,2,3,_) | Put number 9 in the 1st open position from the left. |
| 4 | (x,x,x,10,x,x,x,1) | (9,2,3,10) | Put number 10 in the 1st open position from the left. |

Figure 2

Now using the example (9,2,3,10) we are able to determine the fitness of this member (one possible solution) based on the total cost. To do this we will add up the cost of each link at that time period. The position in the order determines the time period to use in Table 2. The result obtained is shown below.

$$\text{Fitness Value} = \text{Cost}[\text{Link in Position 1, Time} = 1] + \dots + \text{Cost}[\text{Link in Position } x, \text{Time} = x] \dots + \text{Cost}[\text{Link in Position } s, \text{Time} = s]$$

$$\text{Fitness Value} = \text{Cost of Link \# 9 constructed first} + \text{Cost of Link \# 2 constructed second} + \text{Cost of Link \#3 constructed third} + \text{Cost of Link \# 10 constructed fourth.}$$

$$\text{Fitness Value} = \quad 69 \quad + \quad 87 \quad + \quad 39 \quad + \quad 24$$

$$\text{Fitness Value} = 219$$

This is the total cost to build this one possible solution. This is only one member from the total population. We want to begin with 50 members by picking 50 sets of random numbers of 8 position arrays according to the permutation rules. The fitness value of each of the 50 members in our population is then calculated. Once calculated, we order them from lower to higher fitness. Remember lower fitness means it is cheaper to build the links. We take the top 20 and mate them using crossover and we take 10 mutations until we have another population of 50. This process is repeated a number of times until the fitness does not change for 50 generations or until the best value (determined from comparison to the Brut Force) is determined. The population size (50) and the population size we keep (top 20) can be adjusted to test the sensitivity of the Genetic Algorithm as described below.

6. Sensitivity of the GA

This study will perform three objectives in order to measure the sensitivity of the genetic algorithm:

1) Sensitivity of genetic algorithms with respect to how the cost actual changes with time

An important point of interest here is what affect varying the cost changes with each time point has on the number of generations needed to obtain the most fit solution. The cost changes will be varied in the following manner:

- a) Completely random (RA)– A cost value for each time period will be generated using a random number generator.
- b) Non decreasing (ND)– At each time period, the cost will increase a few percent from the cost of the previous time period.
- c) Non increasing (NI) – As each time period progresses, the cost will decrease a few percent from the cost of the previous time period.
- d) Highly Multimodal (MM) – As each time period progresses, the percent change will vary. Also from one time period to the next the value can increase as well as decrease however, the percent of change from one time period will be fixed.
- e) Randon Multimodal (RM) – As each time period progresses, the percent change will vary. Also from one time period to the next the value can increase as well as decrease however, the percent of change from one time period will vary.

Analysis of 30 different cost tables will be run for each variant described above. An analysis will then be performed to determine what effect, if any, varying the cost will have on the number of generations needed to obtain the most fit solution.

2) The sensitivity of genetic algorithms based on the size of the encoding

Varying the number of links in turn varies the length of the encoding. This will in turn vary the length of the permutation code. It is interesting to see what effect the length of the code has on the number of generations needed to obtain the most fit solution. To measure this we will take problems with 4 links, 6 links and 8 links. We will call the number of links in the problem S. Thirty variants will be taken for each value of S. An analysis will then be performed to determine what effect, if any, varying the size of the encoding will have on the number of generations needed to obtain the most fit solution.

3) The sensitivity of genetic algorithms based on the population size and the mutation rate.

In this exercise, the population size is defined as the size of the population before any crossover or mutation has occurred. The generation size (size) is defined as the size of the population after all crossover and mutation is complete. When the grim reaper selects the more fit solution, the population returns to the population size and the procedure repeats again. The table below shows the different population sizes and Generation sizes (size) that were tested.

| Population Size | Generation Size (Size) | Number of Parents | Number of Offspring Produced per Generation |
|-----------------|------------------------|-------------------|---|
| 50 | 100 | 50 | 50 |
| 50 | 200 | 50 | 150 |
| 100 | 200 | 100 | 100 |
| 150 | 200 | 150 | 50 |

Table 3

In addition a 30, 60, and 90 percent mutation rate was also studied. In studying the effect of population size the mutation size was kept at 10 percent. When studying the mutation rate the population size was kept to 50 and the Generation size (size) was kept to 100.

Thirty such variants will be taken for each population size and each percent mutation described above. An analysis will then be performed to determine what effect, if any, varying the population size and the percent mutation rate will have on the number of generations needed to obtain the most fit solution.

7. Experimental Results

With each run, the best value was calculated by performing a Brute Force analysis. The genetic algorithm was allowed to run as many generations until it reached the best value determined by the Brut Force value. Occasionally, the genetic algorithm would come close to the best answer but not obtain it until it ran for several thousand generations. This is shown in Figure 3 below. In such cases the genetic algorithm will be forced to stop if the fitness has not changed after running for 50 generations. In each of these cases separate analysis will be done using that value for generation number where leveling had occurred vs. total number of generations to completion.

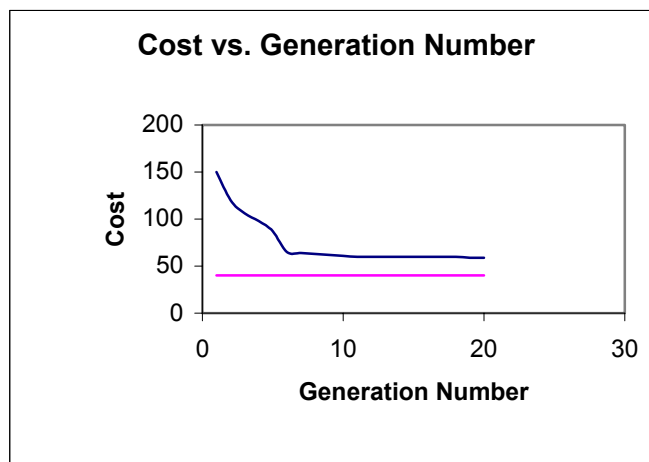


Figure 3

When analyzing the sensitivity of the cost table we find that the more random the cost table the greater the number of generations needed to complete. Figures 4 and 5 contain the results described above. Figure 5 is the same as Figure 4 but accounts for the adjustment described above due to leveling. There is a direct relationship between the number of links and the number of generations - as one increases, the other also increases. Please also note that when we go from a multimodal cost table to a random multimodal cost table, we add some randomness to the cost table. As can be seen that increases the number of generations needed to find the best result slightly.

In Figure 6, we show that the 60 and 90 percent seems to require less generations to find the best result. Figures 7 and 8 shows that the size of 200 seems to reduce the number of generations needed to find the best result although the effect is small.

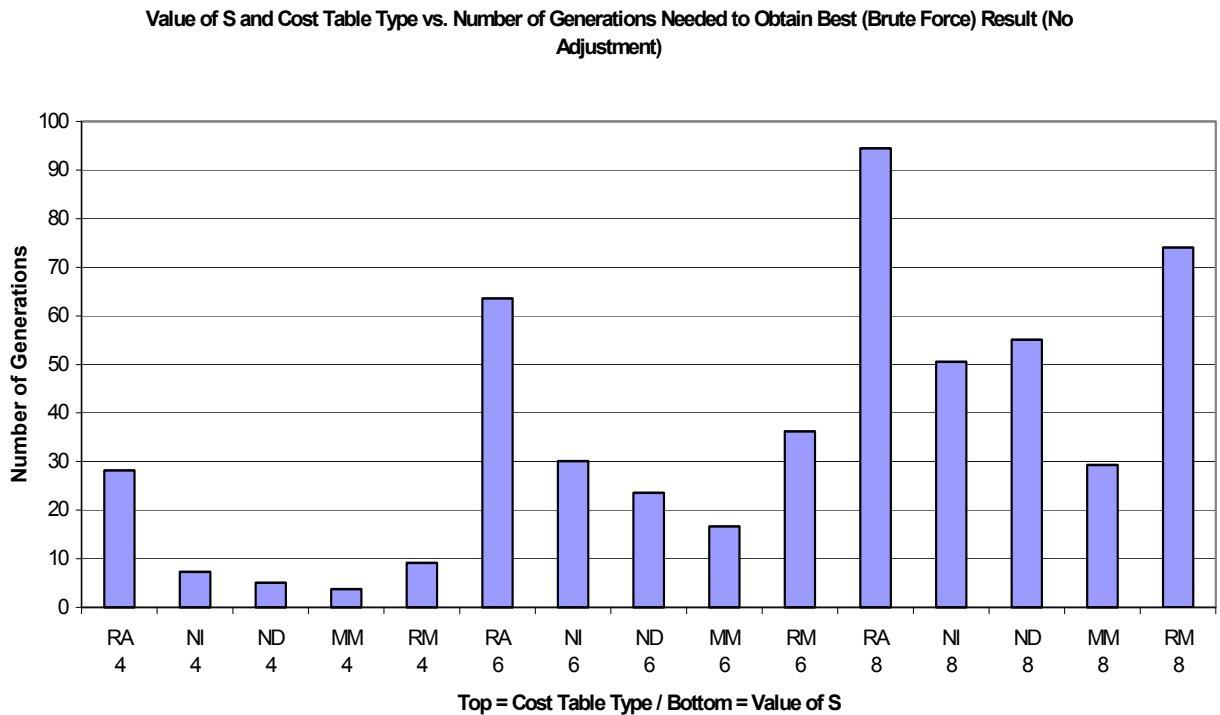


Figure 4

Value of both S and Cost Table Type vs. Number of Generations Needed to Obtain Best (Brute Force) Result
(Adjustment for forced Stopped Results)

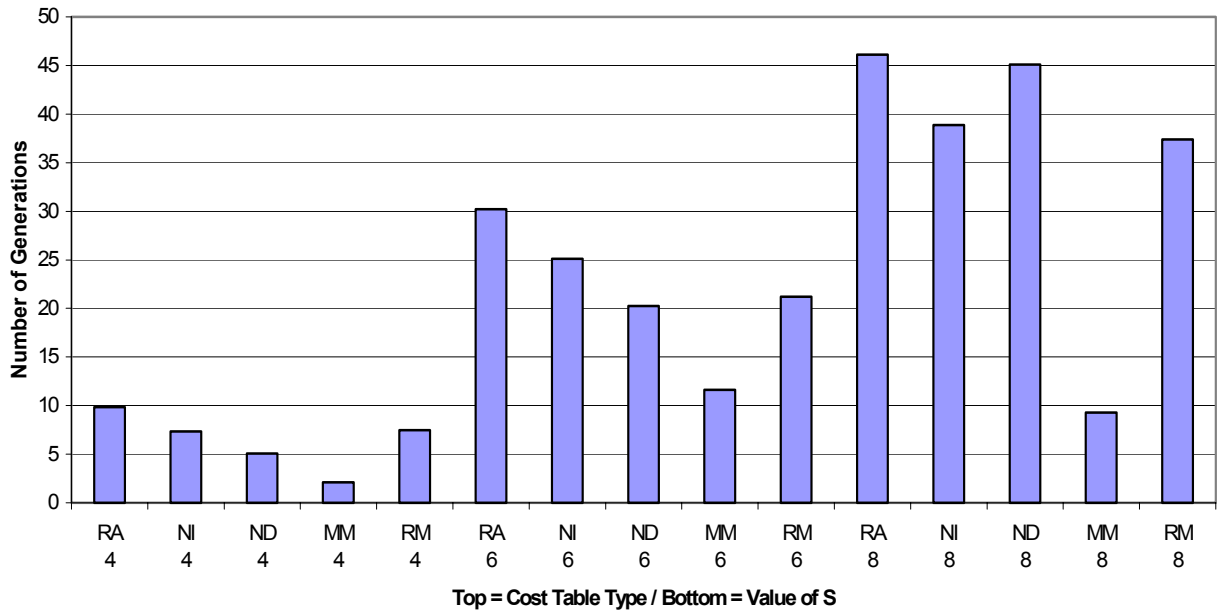


Figure 5

Percent Mutation Rate vs. Number of Generations Needed to Obtain Best (Brute Force) Result

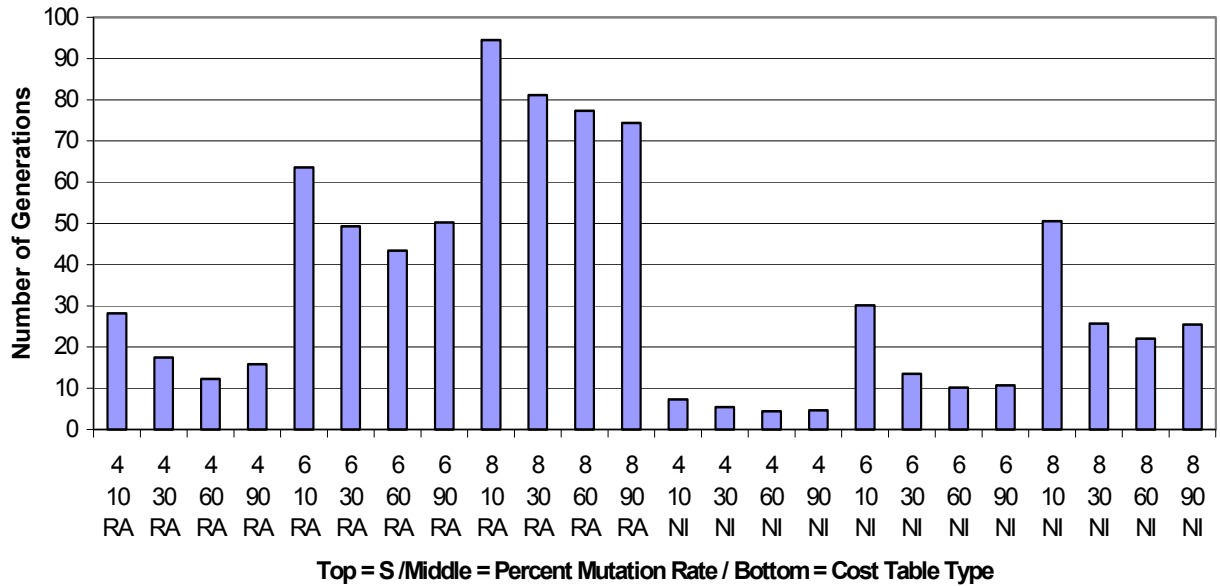


Figure 6

**Population Size vs. Number of Generations Needed to Obtain Best (Brute Force)
Result
Random Access Cost Table**

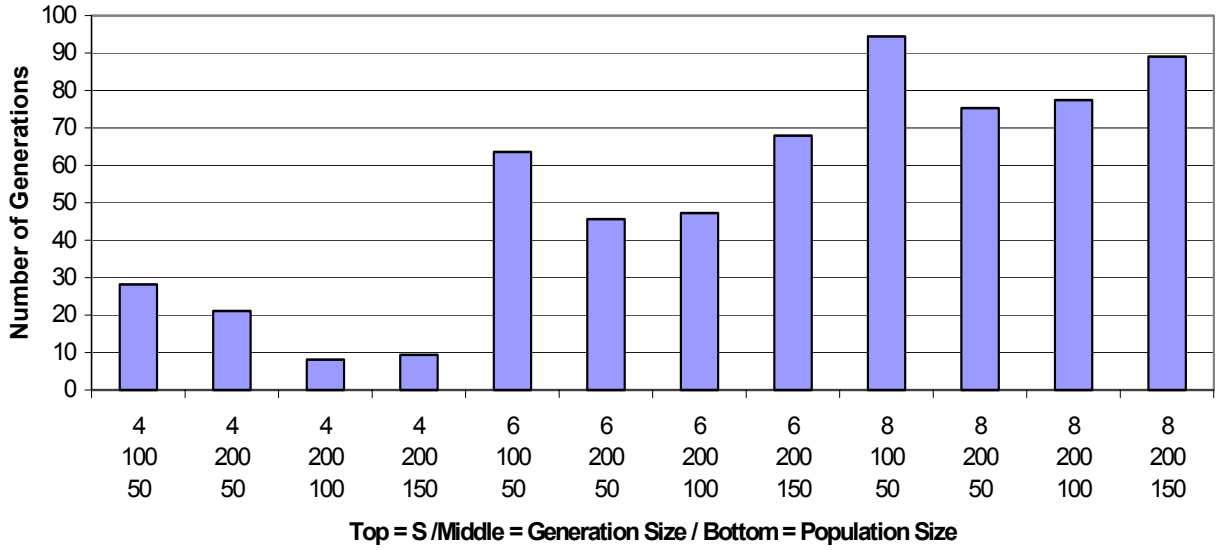


Figure 7

**Population Size vs. Number of Generations Needed to Obtain Best (Brute Force)
Result
Non Increasing Cost Table**

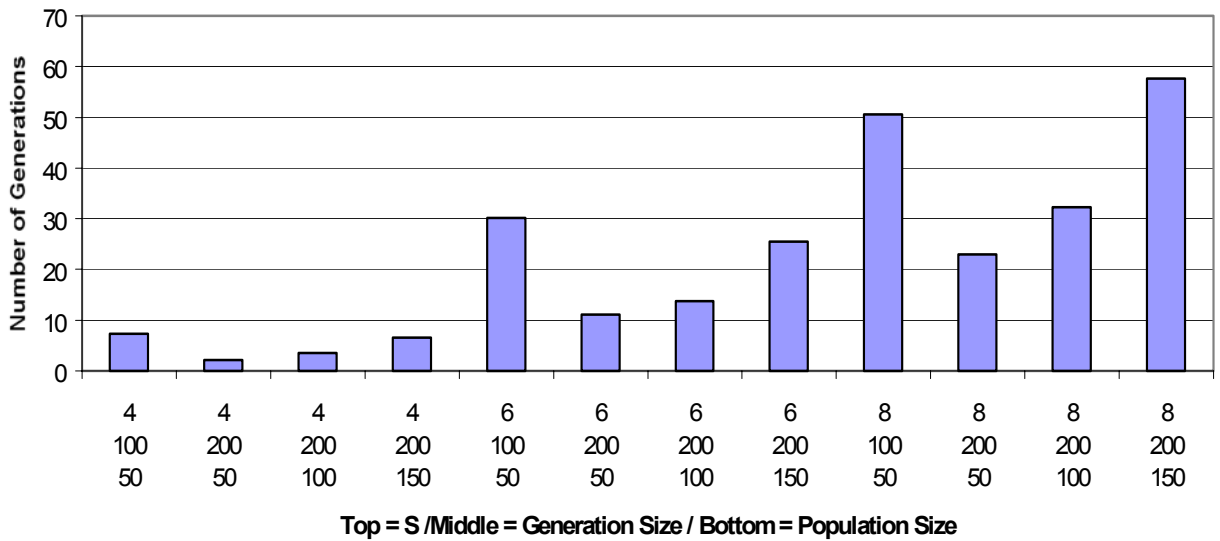


Figure 8

8. Conclusions

The results support the following conclusions:

- 1) The more random the cost table, the greater the number of generations needed to obtain the best value.
- 2) As the size of the encoding increases, so does the number of generations needed to obtain the best value
- 3) Generally, as you increase the percent mutations, the number of generations needed to obtain the best value decreases. This does however, reach some threshold where it has a negative effect.
- 4) A Generation size (size) of 200 appears to require a fewer number of generations needed to obtain the best value than a Generation size (size) of 100.

9. Future Work

The model described above can use dummy slack time periods. I have not implemented that here but plan to do so in future studies.

10. Acknowledgements

I would like to thank Michael L. Gargano and William Edelson for supporting me in this research and for their assistance in conducting this research. I would also like to acknowledge Pace University's School of Computer Science and Information Systems for giving me the opportunity to present this research.

11. References

- 1) Gargano, M.L. and W. Edelson, Optimal Sequenced Matroid Bases Solved By Genetic Algorithms With Feasibility Including Applications, *Congressus Numerantium* 150, (2001) pp. 5-14.
- 2) Edelson, W. and M. L. Gargano, Minimal Edge-Ordered Spanning Trees Solved By a Genetic Algorithm with Feasible Search Space, *Congressus Numerantium* 135, (1998) pp. 37-45.
- 3) Hartsfield, N., G. Ringel, *Pearls in Graph Theory*, Academic Press, pp. 94 - 99.
- 4) Wilson, R.J., *Graph Theory*, Fourth Edition, Addison Wesley Longman Limited, (2000)
- 5) Goldberg, D.E., *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison Wesley, (1989).
- 6) Davis, L., *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, (1991).
- 7) Gargano, M.L. and S. C. Friederich, On Constructing a Spanning Tree with Optimal Sequencing, *Congressus Numerantium* 71, (1990) pp. 67 – 72
- 8) Gargano, M.L., L. V. Quintas and S. C. Friederich, Matroid Bases with Optimal Sequencing, *Congressus Numerantium* 82, (1991) pp. 65 - 77.
- 9) Roberts, F.S. *Discrete Mathematical Models*, Prentice-Hall Inc.,(1970)
- 10) Hillier, F.S. and G. J. Lieberman, *Introduction to Operations Research*, Holden-Day Inc. (1968).
- 11) Rosen, K.H. *Discrete Mathematics and Its Applications*, Fourth Edition, Random House (1998).
- 12) Michalewicz, Z., Heuristics for Evolutionary Computational Techniques, *Journal of Heuristics*, vol. 1, no. 2, (1996) pp. 596-597.
- 13) Edelson, W. and M. L. Gargano, Feasible Encodings for GA Solutions of Constrained Minimal Spanning Tree Problems, late breaking paper at the 2000 Genetic and Evolutionary Computation Conference, 7/00, pp. 82 - 89.
- 14) Mitchell, M. *An Introduction to Genetic Algorithms*, The MIT Press,(1999)
- 15) Haupt, R and Haupt, S.E. *Practical Genetic Algorithms*, Wiley-Interscience,(1998)