N90-25505

Mobile Transporter Path Planning

Paul Baffes and Lui Wang

Artificial Intelligence Section, FM72
Johnson Space Center, Houston, Texas 77058

1. ABSTRACT

(*The work presented here is a copy of a presentation given at the SPIE cambridge symposium in November 1988*). The use of a genetic algorithm for solving the mobile transporter path planning problem is investigated. The mobile transporter is a traveling robotic vehicle proposed for the space station which must be able to reach any point of the structure autonomously. Elements of the genetic algorithm are explored in both a theoretical and experimental sense. Specifically, double crossover, greedy crossover, and tournament selection techniques are examined. Additionally, the use of local optimization techniques working in concert with the GA are also explored. Recent developments in genetic algorithm theory are shown to be particularly effective in a path planning problem domain, though problem areas can be cited which require more research.

2.INTRODUCTION

The current design of the proposed space station involves a rather large, skeletal truss structure to which various living modules, laboratories, and equipment will be attached (see Figure 1). To facilitate the performance of maintenance tasks and the transportation of materials around the station, a mobile transporter (MT) system is being designed which will be capable of traversing the truss structure. At the very least, as the MT moves it must be able to avoid collisions with the objects attached to the truss, and it is obviously desirable that a path of shortest distance be selected before moving the MT between any two points. Since the configuration of the space station is modifiable and the MT must be able to begin its trek from any point on the truss structure, the selection of a good path for the general case is a difficult task.

Planning a path for the mobile transporter falls into a category of optimization problems known as trajectory planning problems. Typically the solution spaces for these problems are multimodal, i.e., they are characterized by many local maxima at which acceptable solutions may be found. Classical gradient techniques for solving such optimization problems have not proven effective due to difficulties in selecting relevant features from the domain from which to build a hill climbing algorithm. Several studies such as one conducted by Gomez-Tierno[1] have suggested that genetic algorithms (GA) offer more promise due to the algorithm's global approach to problem domains. These studies also suggest that more theoretical work on GAs needs to be done before the technique can be satisfactorily applied to trajectory planning. However, recent advances in GA theory have directly addressed problems, such as path planning, whose solutions are order-dependent. What follows is an examination of the application of current GA theory to the mobile transporter problem. The following specific issues are addressed: (1) the details of casting the mobile transporter problem into a genetic algorithm format, (2) predictions of pitfalls or enhancements which can be derived from GA theory, and (3) discussions of some issues still open for using a GA to perform path planning for the mobile transporter.

3. BACKGROUND

3.1 GENETIC ALGORITHMS

It is assumed that the reader is familiar with the various elements of the genetic algorithm and an introduction is not provided here. However, a short review of some of the theory and terminology of GAs is presented as background material for the sections which follow. Recall that the recombination process of a GA typically involves two operators: *crossover* and *mutation*. A crossover occurs between two solutions, called *parents*, which are probabilistically selected from the

PRECEDING PAGE BLANK NOT FILMED

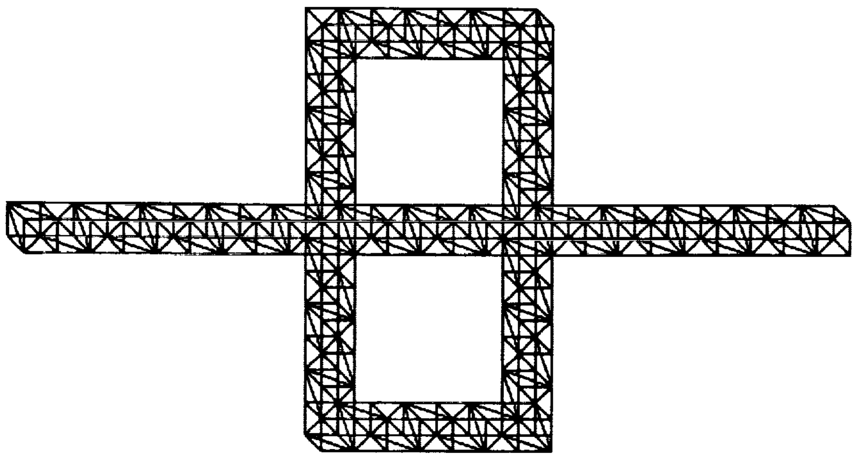

Figure 1. Empty space station truss structure

population based upon their fitness values. A *crossover point* is chosen at random between zero and the length of a solution string, and the two parents are split at that point into front and back ends. The back ends of the two parents are then interchanged and reconnected, creating two new solutions which have roughly half of their values from each parent. Mutation, on the other hand, occurs within a single parent and operates locus by locus. Should a mutation occur at a particular locus point, a new allele value for that point is randomly chosen. The result is a new solution which differs only slightly from the original. While other genetic operators are sometimes used, these two represent the minimum necessary for the correct operation of a GA and are the most widely used.

Theoretical formulations of genetic algorithms have been developed around the concept of a *schema* which represents a partition of the solution space. Roughly speaking, a schema can be thought of as a substring of a solution with some elements given specific values and some left undetermined. For example, given a binary set of allele values (i.e. 0 and 1) a typical schema might look like the following:

*10***1**

where the "*" symbol indicates that either allele value may occur at the given locus. Consequently, the schema may be thought of as representing groups or hyperplanes of substrings which specify particular patterns of allele-locus combinations. Schemata have two important characteristics, their *order* and their *defining length*. The order of a schema is the number of its specified elements; the above schema has an order of 3. The defining length is defined as the distance *between* the outermost specified elements of a schema. In the above example, if we were to number the loci shown we would have 9 loci with the two "1" values occurring at positions 2 and 7. Taking the difference of these yields a defining length of 7 - 2 or 5. Note that this is one less than what might normally be considered the size of the schema, sometimes called the *length* of the schema, which here would be 6. In this paper the defining length will be specified by ∂ (delta) and the length will be specified by D such that $D = \partial + 1$.

Holland's[2] work showed that, given the proper problem domain, genetic algorithms will evolve toward nearly optimal solutions by building up complicated structures out of small components which can be described as schemata. A typical expression of this phenomenon as described by Goldberg[3] would look like the following:

$$n\,(\mathrm{H},t+1) \geq n\,(\mathrm{H},t)\,\frac{f(\mathrm{H})}{\bar{f}}\left[1 - p_c\,\frac{\partial(\mathrm{H})}{\mathrm{L}-1} - p_m\,\mathrm{o}(\mathrm{H})\right] \tag{1}$$

where H stands for a particular schema, n(H,t) is a measure of the number of copies of the schema in the population at time t, f is the fitness function, L is the length of a solution, p_c and p_m are the probabilities of crossover and mutation, and $\partial(H)$ and o(H) are the defining length and order of the schema. The fractional term indicates that the number of schema in the population will change proportional to the fitness of solutions containing the schema relative to the average fitness of the population. Of particular interest is the last term of the equation which is a measure of the extent to which a crossover or mutation is likely to disrupt the schema. Note that the greater the defining length or order of the schema, the greater the chances that it will be altered by a crossover or mutation, thus driving down the number of schema in the population. Consequently, schemata which are *short* and of *low order* are probabilistically favored by the genetic algorithm, and any GA which is to work effectively must contain operators which prefer such small, low order building blocks.

Recent theoretical work by Goldberg and Lingle[4] is centered around the idea of a double crossover technique termed a partially matched crossover (PMX). The PMX operator is particularly effective for domains in which ordering of values is essential. In such problems, both the allele values and their positioning relative to other alleles contribute to the overall fitness of a solution. An example of an ordering problem cited by Goldberg and Lingle is the traveling salesman problem (TSP) in which a salesman must visit some number of cities in a circuit while taking the minimum distance to do so. Crossing over two solutions using the PMX operator begins by selecting two crossover points, instead of just one, which define a *cut* of length K covering all alleles occurring between the two crossover points. The cut regions of the two solutions to be crossed are then exchanged. Any duplications which may result *outside of the cut regions* are swapped between the two new solutions to enforce the constraint of exactly one occurrence of each city. Note that no swapping is allowed to occur inside of either cut, thus preserving whatever order existed among the alleles within that region. Goldberg and Lingle present the survival chances of a schema using the PMX as

$$P(S) = \frac{K-D(H)+1}{L} + \frac{L-K-D(H)+1}{L}\left[1 - \frac{K+1}{L}\right]^{o(H)} \qquad (2)$$

where K is the length of the cut, D(H) is the length (not defining length) of the schema, o(H) is the order of the schema, and L is the length of a solution. The two terms of this equation represent the two conditions under which a schema may survive a crossover: entirely within the cut region or entirely outside of the cut region. Schemata which span either of the two crossover points are assumed to be disrupted. The first term of the equation, for schema inside the cut, is only dependent upon the length of the schema. Again, short schema are favored. The second term, for schema outside of the cut, is dependent upon both the length and the order of the schema. The second factor of this term is a measure of the chances that any single locus outside of the cut will survive the swapping phase of the PMX. Since this chance is the same for all specified bits of the schema, the more specified bits the schema has the lower its chances of survival. Thus the lower order schema are also favored by the PMX operator.

To summarize, this review has presented two formulas which show the probability of schema survival. The first covers single crossover cases and the second can be used for the PMX crossover operator. The analysis section which follows (section 4) will discuss how these formulas relate to the application of GAs to the mobile transporter problem.

3.2 THE MOBILE TRANSPORTER

Several features unique to the mobile transporter's design and operating environment are salient to the use of genetic algorithms as a modeling approach. As it is currently envisioned, the MT will be able to transport heavy loads (up to 300,000 lbs.), which will substantially hinder its rate of travel. It is estimated that the travel time between the two most remote points of the station may be as much as 6 hours. Furthermore, since the truss structure of the space station is 3-dimensional, the MT will have at least two types of movement:

along the surface of one plane, or between two planes. The latter is a much more complex maneuver lasting approximately five times the duration needed to travel an equivalent distance along a single plane. Due to these constraints, path planning for the MT must be efficient and adaptable so that changes in the scheduled uses of the MT may be quickly addressed by an appropriate change of path.

However, trajectory planning for the MT does not map directly into a genetic algorithm format. One of the first obstacles encountered is the need for a *variability of solution length* which does not occur in a classical GA design. The paths developed for the MT vary in length not only between problems as the MT is assigned different starting and ending points, but also within a single problem as the GA is running. Furthermore, the existence of variable length solutions complicates the structure of the crossover operator, since some strings will have more crossover opportunities than others and since randomly chosen crossover points will almost certainly produce invalid solution paths. To ensure that a crossover produces a valid result, one can use an *intersection crossover*. An intersection crossover is identical to the traditional crossover except for a preceding step to compute the set of intersecting points for the two parents to be crossed. Once calculated, this set is used as the basis for the selection of a crossover point, thus ensuring that the substrings of the two parents used to make the new solution will be rejoined at a common point. Note that the use of the intersection crossover has two potentially adverse effects: the selection of a crossover point is no longer a uniformly random choice (those points that are more likely to be present in a path will be chosen more often) and the probability of a crossover occurrence is diminished because some parent pairs may have an empty intersection set. Finally, observe that double crossovers performed using the set of intersecting points can produce cycles in the resulting solutions, even if the parent paths of the initial population of solutions are constrained to be free of cycles. Such loops in the resulting solutions must be removed if the constraint of no-cycle paths is to be maintained in subsequent generations.

A *delooping* procedure which will accomplish this can be defined as follows. Starting with the beginning of the path, a candidate point is chosen and all points following the candidate are visited in turn. If any of these following points matches the candidate, all elements of the path between the candidate and the matching point are removed from the path. The checking then continues with the same candidate until the end of the path is reached, at which time the point currently following the candidate is chosen as the new candidate. The process is repeated for all points until the end of the path is reached. Note that no more than two matches will be found for any given delooping operation. This is due to the fact that both the original path and the region spliced in from the other parent are guaranteed to have no cycles. As a consequence, no two matches in the crossed over solution will be identical and all such matches will occur between a point inside of the cut region and a point outside of the cut region. Since delooping removes everything between two matched points, a single delooping action must cross one of the two crossover points thereby removing it from the solution. With only two crossover points in a double crossover, only two such delooping actions can occur.

Thus, several features of the mobile transporter path planning problem make it incompatible with classical GA techniques. In particular, the existence of varying length paths gives rise to a need for a different mechanism for the crossover operator. An intersection crossover is defined to ensure that crossing over will produce valid paths, but this in turn gives rise to a cycling problem when two crossover points are used. Consequently, a delooping procedure has also been developed to rid a path of cycles. The resulting crossover operation becomes a combination of intersection crossover followed by the delooping procedure. However, a question remains open as to the possible adverse effects of these new operators on the functioning of the resulting GA.

4. APPROACH

A twofold approach was taken to investigate the translation of the mobile transporter problem into the format of a genetic algorithm. First, an examination of the current GA theory was made in an effort to predict the behavior of the resultant system. Second, several simulations were written to collect experimental data on the various GA methods under question. The focus of these two approaches was aimed at the following topics:

1. *Single vs double crossover.* It was felt that the MT path planning problem was significantly order-dependent, but the degree to which this was true was unknown. It was questionable whether a single crossover would be sufficient for a problem which exhibited only a small degree of dependence upon the order of its values and, conversely, whether or not a double crossover in such a situation would be disruptive.

2. *Effects of intersection crossover and delooping.* Given that a new crossover operator was needed for performing double crossovers, it was not immediately clear what effect this would have on the GA's performance. The question arose as to whether the new crossover operator might adversely effect the diversity of the population due to some points in the domain being more likely candidates as crossover points.

3. *Use of a greedy crossover.* Liebens et al[5] describe a process whereby a greedy algorithm technique was employed during the selection of crossover points which greatly enhanced the conversion of the GA to an optimal answer. The extent to which this idea would help in the MT problem was an intriguing question.

4. *Need for a mutation operator.* Because the GA described here used a delooping procedure rather than the swapping procedure of the PMX, it was unclear to what extent this might effect the preference for low order schema. Recall the presence of an order-dependent factor in the second term of equation (2). Since swapping was replaced by delooping, this factor would not present in the GA for the mobile transporter, resulting in the loss of the only order-dependent term of the equation. If a mutation operator could be devised which would return such a factor to the equation, the question then became whether to allow mutations to occur within the cut region of the double crossover or to force all such.changes to occur outside the region as is done with the PMX operator.

5. *Local optimization.* During our simulations, we found that the GA seemed to converge before the last few "kinks" of a path could be straightened out. Consequently, our analysis included an examination of the extent to which local optimization techniques could be used to enhance the overall performance of the GA.

6. *Tournament selection.* Normally, parents are chosen for recom-bination based on a roulette-like approach where each parent occupies a percentage of an imaginary roulette wheel proportional to its fitness. Parents with the highest fitness values are chosen most often. However, this method of selection has two disadvantages. If fitness values vary widely, a super-fit parent can easily dominate the population and drive the GA to premature convergence. ON the other hand, if fitness values are clustered, small differences which may be important are not emphasized enough. Tournament selection provides an alternative to roulette selection which avoids both of these problems.

5. EXPERIMENTAL RESULTS

Much of what was expected was borne out by the simulation results. Three different simulations were developed to test the issues cited in section 4 above: a simulation of the mobile transporter, a reproduction of the experiment performed by Goldberg and Lingle for the TSP problem, and a simple scheduling simulation. The results for crossover operators are presented first, then the tests involving the mutation operator, and finally the results of using local optimization and tournament selection.

5.1 Crossover

Figure 2 shows a comparison between typical single and double crossover runs for the mobile transporter genetic algorithm. The crossover technique used for this simulation was the intersection crossover followed by the delooping procedure described in section 3.2. Note that the descending fitness values of subsequent generations indicate that the path length is improving by getting shorter. As expected, the MT problem exhibits enough order-dependence to benefit from the use of a double crossover.

Moreover, the double crossover proved to have no adverse effect in the simple scheduling problem. In this simulation, four fictitious employees were to be scheduled over a three week period with one employee working per day. Each employee was required to work between some minimum and maximum number of days, with some employees working less than others. Also, no employee was allowed to work two days in a row. The only ordering constraint introduced into the problem was a requirement that one particular employee always work before another employee each week, though no specification was made as to how far apart the two employees had to work. Fitness values were determined by counting the number of constraints violated by a solution, thus lower fitness values represented better solutions (fewer constraint violations). Table 1 shows the average fitness of 96 runs using both single crossover and double crossover GAs to solve the problem. Note that each GA was

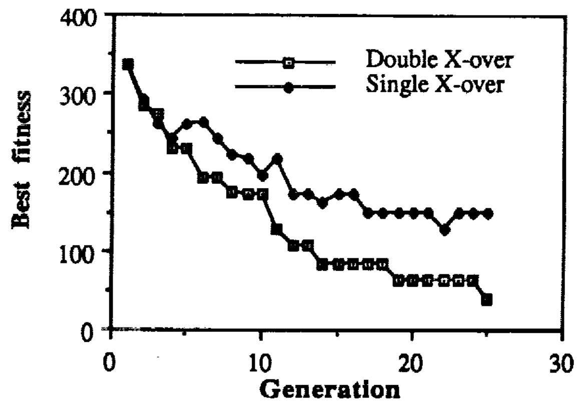

Figure 2. Best fitness for single/double crossovers

allowed to run until it converged on a single fitness value for all members of its population. While Table 1 may not show any significant advantage gained by using a double crossover, it is clear that nothing was lost by its use, as was expected.

Lastly, Figure 3 shows that the greedy crossover method proved to be an excellent choice. To implement a greedy crossover, the double crossover operation was altered to select from among a given set of intersection points the two elements which would create the shortest resulting possible solution. The result produced only one child for each parent pair, thus twice as many crossover operations were performed per generation. By adding a greedy method to the crossover, the GA tended to find equivalent path lengths 5 to 10 generations earlier than was possible without the greedy crossover. Even though the computation needed to perform a crossover increased, the overall execution of the GA was lessened by lowering the number of generations required to solve the problem.

Table1. Crossover comparison for simple scheduling problem

Crossover used	No. runs	Average fitness
single crossover	96	14.2
double crossover	96	11.0

Table2. Mutation operators and the PMX crossover

Mutation used	No. runs	Best fitness	Average fitness
no mutation	44	1266	1311.2
limited mutation	44	1240	1291.8
unlimited mutation	44	1224	1283.9

5.2 Mutation

The consequences of using the mutation operator were more difficult to characterize. Two different types of simulations were run which tested mutations with both constant and variable length strings. The first simulation, built using the PMX operator and the swapping mutation described in the latter part of section 3.1, was run on a traveling salesman problem covering 40 cities. Three types of runs were made: one with no mutation, one with a limited mutation which was allowed only outside of the cut region, and one with an unlimited mutation which could occur anywhere in the solution. All runs were allowed to converge and a total of 44 runs of each type were recorded. Table 2 shows the results of these simulations. Though the prediction of better performance using a mutation that could effect the inside of a cut region was shown to be true, its effects were not as large as expected. This may have been due, in part, to the fact that relatively small population sizes were used (30 to 50 members), but the extent to which population size has an effect upon mutation was not investigated. More statistical results will have to be presented on such data. The conclusion drawn here was that mutations which could occur inside the cut region certainly had no adverse effects upon the order-dependence of the solution strings, and may have enhanced the search for a more optimal solution.

The second simulation, in which an attempted was made to translate the benefits of mutating inside the cut region to the MT genetic algorithm, proved to be fruitless. A swapping mutation operation such as the one used with the PMX crossover could not be used since the result would, in general, produce an invalid path. In fact no single-locus mutation operator could be devised. The scheme finally adopted was to select a cut region similar to the double crossover cut by randomly picking two points of the solution string. All elements between the mutation points were removed and a new random path was generated to bridge the gap. The mutation probability also had to be changed since it now referred to the whole solution rather than each locus individually. With each solution averaging approximately 50 elements, this yielded about a 5 percent (50 x 0.001 for each locus) total chance that any given string should mutate. Figure 4 shows that the use of this mutation operator had no effect on the performance of the mobile transporter GA.

Part of the reason for this poor performance may be explained by the fact that the mutation operator chosen was not dependent upon the order of a schema. By choosing a *region* for mutation, rather than some point-by-point operation, schema which survive the mutation process will most likely be short in length. But this is exactly the situation already enforced by a double crossover, thus no further preferences result from the use of the mutation operator. Consequently, as is shown by Figure 4, nothing is gained by adding the mutation operator. However, it is important to note that this is only true for the mutation operator defined here; it may be possible to define other mutation techniques which would have a beneficial

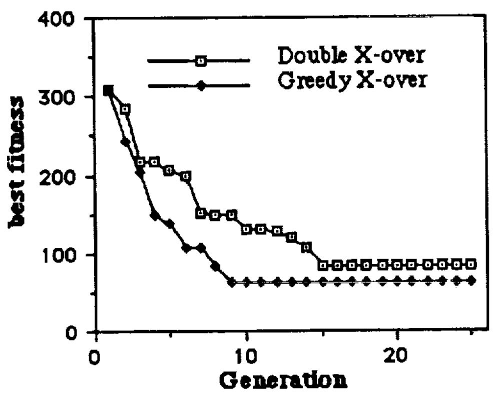

Figure 3. Double and greedy crossovers

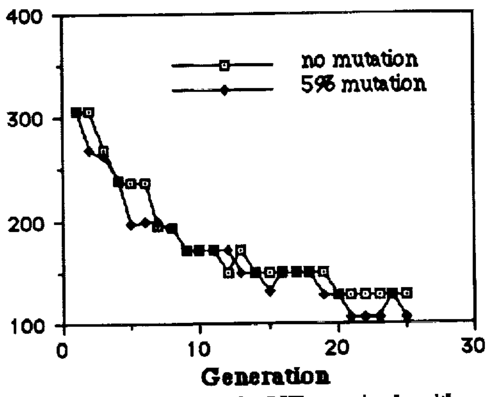

Figure 4. Mutation in the MT genetic algorithm

effect on the GA.

5.3 Local Optimization

Without an effective mutation operator which could keep the diversity of the population high, it became important to look for a means of directing the GA more quickly towards an optimal solution. Both the local optimization method we employed and the use of tournament selection helped in this regard. Local optimization was implemented by taking advantage of the inherent geometric regularities of the Space Station truss structure. Referring back to Figure 1, note that large sections of the truss structure form contiguous planes. These planes can be thought of as single units or "blocks," broken apart only when some obstacle attached to the truss structure divides the plane, or when tone plane intersects another. Rather than forming a path point by point, one can work block by block, where a block represents a contiguous length of unbroken space along some plane of the truss structure.

Figure 5 is a dramatic example of the effects of using this local optimization technique on the MT path planner problem. Not only does the average fitness of the population start out much better, but the overall performance of the GA is twice as good by the time the population converges to a single answer. In some sense this is not too surprizing since the optimization technique used greatly reduces the search space for the GA. However, it is clear that such techniques can enhance the final output of a GA when features of the problem

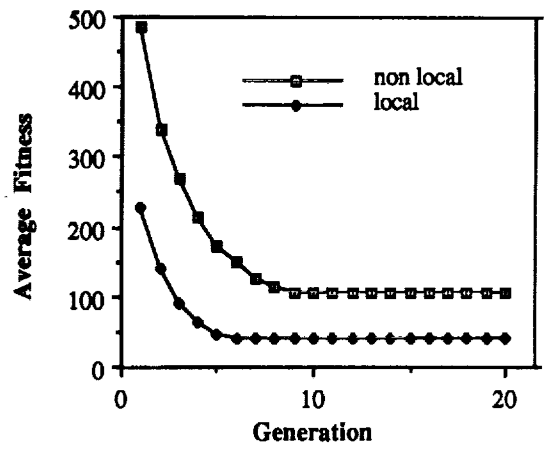

Figure 5. Effects of local optimization

domain permit their use.

Finally, Figures 6 and 7 illustrate a comparison of roulette and tournament selection methods. Tournaments of 2 were used for these experiments. As expected, tournament selection slows the rate at which a super-fit parent is allowed to proliferate through the population. The result is a higher diversity for a longer period of time, as can be clearly seen in Figure 7. Because no mutation operator could be designed for the MT path planner, this prolonged diversity became important to the overall performance of the GA. Without tournament selection, the GA converged upon a solution too quickly, missing better solutions latent in the population.

6. SUMMARY

The genetic algorithm approach proved to be an effective means for quickly solving

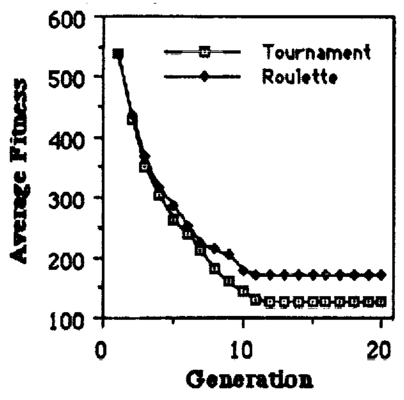

Figure 6. Tournament avg. fitness

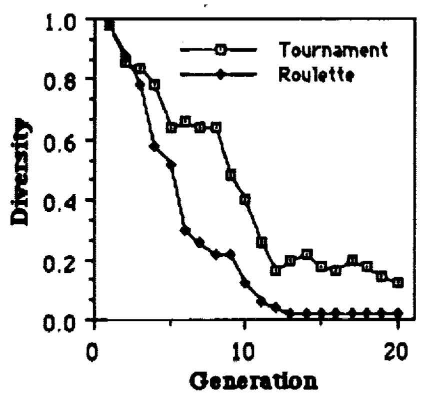

Figure 7. Tournament diversity

the mobile transporter path planning problem. Even though the MT problem did not map directly into a GA format because of the variable length of its solutions, a GA was successfully applied which produced optimal results within relatively few iterations of the algorithm. This GA used a greedy double crossover, tournament selection, local optimization which grouped contiguous points of the truss structure into "blocks," and a delooping procedure to remove cycles from offspring. The greatest improvement in performance came with the addition of local optimization and greedy crossover. Neither the presence of the delooping procedure, nor the fact that some points of the domain space were more likely to be chosen as crossover points, had any adverse effects upon the efficacy of the algorithm. No mutation operator was used since none could be devised to effect solutions on a point-by-point basis. More research still needs to be done on both the mutation operator and the effects of using an intersection set for the choice of crossover points.

Ultimately, an operational solution to the MT path planning problem may be implemented using an exhaustive coding technique due to the relatively small size of its problem domain. However, since trajectory planning is a common problem in space systems which have much larger domains, the genetic algorithm will remain an attractive alternative to the classical techniques used to solve such problems.

7. ACKNOWLEDGMENTS

The authors would like to thank Dr. David Goldberg and Robert Smith of the University of Alabama for their explanations of genetic algorithm techniques and their helpful discussions.

8. REFERENCES

1. M. A. Gomez-Tierno, Study on the use of the genetic algorithm for the solution of global optimization problems, Government Report #GMV-204/85-V2/85, 151 pages (1985).

2. J. H. Holland, Adaptation in Natural and Artificial Systems, University of Michigan Press, Ann Arbor (1975).

3. D. E. Goldberg, "Simple genetic algorithms and the minimal, deceptive problem," in Genetic Algorithms and Simulated Annealing, Lawrence Davis, ed., Morgan Kaufmann Publishers, Inc., Los Altos, 74 - 88 (1987).

4. D. E. Goldberg and R. Lingle, "Alleles, loci, and the traveling salesman problem," in Proc. Intl. Conf. on Genetic Algorithms and their Applications, 154 - 159 (1985).

5. G. E. Liebens, M. R. Hilliard, M. Palmer, and M. Morrow, "Greedy genetics," in Proc. Intl. Conf. on Genetic Algorthims and their Applications, 90-99 (1987).