Proceedings of the International Conference on Information and Automation, December 15-18, 2005, Colombo, Sri Lanka. 1

A Genetic Algorithm Approach to Solve the Shortest Path Problem for Road Maps

Sachith Abeysundara*, Baladasan Giritharan[+], Saluka Kodithuwakku[◊]
*Department of Statistics and Computer Science,
Faculty of Science, University of Peradeniya, Sri Lanka
Email: sachith@email.com
Telephone: (+94) 81 2374652
[+] Department of Statistics and Computer Science,
Faculty of Science, University of Peradeniya, Sri Lanka
Email: bgiri@pdn.ac.lk
[◊] Department of Statistics and Computer Science,
Faculty of Science, University of Peradeniya, Sri Lanka
Email: salukak@pdn.ac.lk
Telephone: (+94) 81 2394260

***Abstract*—This paper presents a new genetic algorithm approach to solve the shortest path problem for road maps. This is based on the analogy of finding the shortest possible distance between two towns or cities in a graph or a map with potential connection, which means that the path distances are always positive. Typically this is represented by a graph with each node representing a city and each edge being a path between two cities and there exist some traditional algorithms that produce solutions for the problem. A new method is found to solve the shortest path problem using GAs. The algorithm has been tested for a road map containing more than 125 cities and the experimental results guarantee to provide acceptably good solutions for the given search space.**

I. Introduction

THE shortest path problem is typical in the world of combinatorial systems. This research will attempt to apply a Genetic algorithm to solve this problem based on a real world system. This is based on the analogy of finding the shortest path (i.e. the shortest possible distance) between two towns or cities in a graph or a map with potential connections (assuming that the path distances are always positive). Typically this is represented by a graph with each node representing a city and each edge being a path between two cities. So, applying a genetic algorithm is an interesting idea.

This is clearly different from traditional algorithms that try to compare every possibility to find the best solution, which might be a time consuming algorithm for a graph containing a large number of nodes and edges.

A. Approach to the Problem

Genetic Algorithms (GAs) [1]-[4] are adaptive heuristic search algorithms which are premised on the evolutionary ideas of natural selection and gene types. The basic concept of GAs is designed to simulate processes in a natural system necessary for evolution, specifically those that follow the principles of survival of fittest, first laid down by Charles Darwin. As such they represent an intelligent exploitation of a random search within a defined search space to solve a problem. Basically, several random sets of parameters are applied to an algorithm and a fitness value (optimization value) is calculated for each. Based on this fitness values, the best sets are mixed (this is a combination of Selection, Crossover and Mutation) together and new sets are again applied to the algorithm until an optimal parameter(s) are obtained. This effect is usually obtained by breaking the genetic algorithm into a few small parts.

This research will use a genetic algorithm that attempts to optimize the parameters of the Shortest path problem, Again, optimizing will most likely mean finding an answer that is consistently accurate or satisfactory, though perhaps not the perfect answer to the problem. The goal is to really find some generally good parameters for the shortest path problem.

B. Outline

Methodology: A detailed Description about the approach and the final design is discussed here.

Results and Discussion: Results obtained from the research is given and discuss how the variations of the parameters which affect the end result.

Conclusion and Recommendations: Content of the research with further extensions is concluded.

II. METHODOLOGY

A. Analyze the Problem

The research problem is different from the traditional TSP (Traveling Salesman problem) since the latter mention case the resulting tour should contain all the cities with no repetitions while that of the shortest path problem only contains only the cities to be traversed through the shortest possible route.

So it is difficult to map the Shortest path problem with Traveling salesman problem as problems with similar constraints.

At the beginning the TSP is taken to identify and to understand the concepts and constraints of the research goals. Earlier, a simpler version of a genetic algorithm was designed for a map with few towns. Later some modifications were done to achieve the successful results.

Herein a detailed description of the Genetic Algorithm parameters is given.

B. Design

Search space representation: Each town in the map is given a unique integer value index from 1……N, where, N is the number of cities in the map.

Individual: Each individual is designed to represent a solution for the problem and it should not contain repeated town indices. The length of the individual is chosen to be equal to the number of towns in the total map, because there may be cases such that the shortest path may contain the total number of cities which may be similar to the result in TSP.

e.g. For a map with N number of cities the gene length is equal to N, where T_i is the i^{th} town in the map

$[T_1, T_2, \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots, T_{(N-1)}, T_N]$

Fitness value is calculated for each according to the given fitness function as well as a rank fitness which is given to each individual considering the fitnesses of the other individuals in the population.

Representing a Chromosome (Encoding type): Chromosome is represented as an Integer array string. Each town index is a gene of the chromosome and the number of towns in the path may differ from individual to individual (Fig.1). To keep a track of this, each chromosome is given a variable called NUM_TOWNS_PATH to hold the number of towns in the path.

9	3	1	5	6	-	-	-	-	-

Fig. 1. An example of a Chromosome. A route from town '9' to '6' is represented through the cities 3, 1, 5.

Initial population: The size of the population depends on the number of towns in the Map. For a map with 125 towns it is better to have a population size around 100.

Individual generation for the initial population: For this initial selection the direction of SOURCE to DESTINATION is taken into consideration. Four directions from source to destination are considered and named as NORTH_WEST, NORTH_EAST, SOUTH_EAST, SOUTH_WEST.

The algorithm for selecting an individual for the initial population is given below.

```
adir: direction from SOURCE to DESINATION
dir: direction between two adjacent towns
twn: next town to be visited

Begin
  dir = adir
  Initialize an empty Individual
  Set the 0th index to the SOURCE town
   Do
     S = the set of towns at the direction dir
     While S is empty and more directions to search
        S = get towns at other three directions -
              - according to the distribution of towns
     End while
     If S is empty
      twn = DESTINATION
     Else
      twn = select a town from S
     End if
     If twn not exists in the individual
      Add the twn to the individual
      dir = get direction twn to DESTINATION
     Else
      twn = last town in the individual
     End if
    Repeat until (twn = DESTINATION)
End
```

If the Population size is N then N/2 number of individuals are generated from SOURCE to DESTINATION and the other N/2 are generated from DESTINATION to SOURCE and rearranged.

Empty individual with the length of N is initialized at the beginning. The first index is set to the starting town index. Execute the above algorithm to generate an initial individual. Situations that the algorithm cannot complete a possible tour then complete the tour by adding the destination town to the individual. This results in an individual with negative fitness value according to the given fitness function.

Individual generation is repeated until the population contains N number of individuals.

Fitness function: The fitness function (1) is defined as follows.

$$F(x) = \frac{1}{Actual\_length\_of\_the\_path} - The\_\#\_disconnected\_paths \quad (1)$$

Consider the first part of the function,
Actual_length_of_the_path is calculated by summing the route distances of the path. Therefore, the value for Actual_length_of_the_path is between 0 and 1. According to the Initial selection The_#_disconnected_paths can only get 0 or 1. It means that the function varies from +1 to -1.

If there isn't a path between town t_1 and t_2 then the SLD (Straight Line Distance) between t_1 and t_2 is considered as the actual distance.

Assigning a Rank Fitness: After generating the initial population, fitness values for each individual are calculated and the population is ranked cumulatively (2). The individual which has the X^{th} rank in the population has a rank fitness R_f,

Where,

$$Rf = \sum_{i<X} i^2 \quad (2)$$

This rank fitness is updated at the end of each generation.

C. Next Generation

Elitism: 10% of elitism is used to retain the individuals with best fitnesses for the next generation.

Selection operator: Roulette rank selection is used according to the rank fitness given to the individuals to select two parent individuals for crossover operation.

.

Crossover operator: A multi point crossover is used to breed new children. Two selected parents are crossed over at every point with equal town indices according to a probability P_c [70% - 90%]

Mutation operator: 10% mutation is done to the population individuals. A random position **P** is selected in the individual and then a path from **P** to the destination is re-generated.

.

Termination criteria: If the individual with best fitness is repeated for longer time the algorithm terminates.

Under the above constraints the next generations are derived and it is repeated until the algorithm reaches an optimal solution.

III. RESULTS AND DISCUSSION

A. Results

The implemented algorithm has been tested for a road map containing more than 125 cities and the experimental results guarantee to provide good solutions acceptable for the given search space.

In most situations the initial population may contain routes that are impossible to traverse in practical situations. The selection criteria for the new generations are totally based on the fitness value given to each individual. But the changes in probabilities of recombination operators may produce surprising results. So it is convenient to keep the operators' values in a certain acceptable range. Fig.2 and Fig.3 shows the behavior of the algorithm when crossover and mutation probabilities changes.

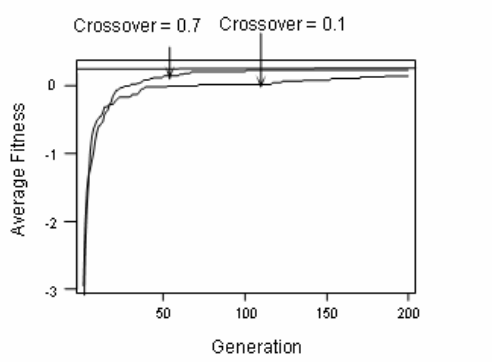

Fig. 2. Different behaviors of the algorithm when the crossover probability varies.

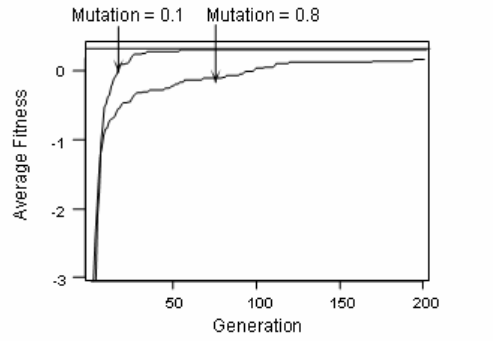

Fig. 3. Different behaviors of the algorithm when the mutation probability varies.

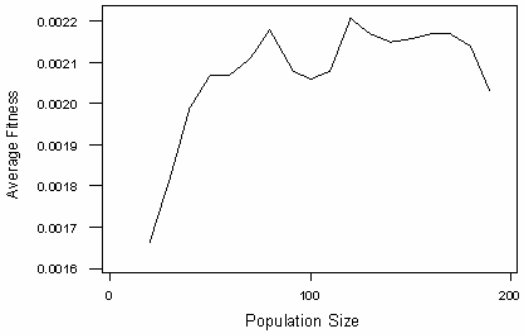

Fig. 4. Above figure shows how the varying size of population affects the final result. Smaller population size may result a slower convergence to the solution as well as the larger values. So it cannot be considered that a larger population size is always good. So this parameter will depend on the size of the search space.

B. Discussion

Major advantage of this algorithm is when it converges to a solution it is able to find another set of possible solutions that the destination can be reached. There may be reasons to ignore the current best path and choose another path, which is similar to the existing one by ignoring some cities of the map. The algorithm is tested for a map with about 125 cities; still the results are produced in an efficient manner. The algorithm tries to converge to a solution according to the generated initial population. So there is less evidence of program to be stuck. This would make the program to run under a large search space with small space complexity. The problem in this algorithm is that according to the initial population, sometimes the individuals may rapidly come to dominate the population, causing it to converge on a local maximum as it is common for many GAs. Once the population has converged, the ability of the GA to continue to search for better solutions is effectively eliminated.

IV. CONCLUSION AND RECOMMENDATIONS

The main advantages of genetic algorithms are their flexibility and robustness as a global search method. They are "weak methods" which do not use gradient information and make relatively few assumptions about the problem being solved. They can deal with combinatorial problems with NP hardness as well as problems with multiple local optima. They are also readily amenable to parallel implementation, which renders them usable in real-time. The primary drawback of genetic algorithms results from their flexibility. The design should come up with encoding schemes that allow the GA to take advantage of the underlying building blocks. One has to make sure that the evaluation function assigns meaningful fitness measures to the GA. It is not always clear how the evaluation function can be formulated for the GA to produce an optimal solution.

GAs are also computationally intensive and convergence is sometimes a problem. One thing that is striking about genetic algorithms and the various parallel models is the richness of this form of computation. What may seem like simple changes in the algorithm often result in surprising kinds of emergent behavior. Recent theoretical advances have also improved the understandability of genetic algorithms and have opened the door to using more advanced analytical methods.

The research is to propose a new method to solve the shortest path problem using genetic algorithms. This solution aims to achieve an increased number of successful and valid convergence using evolutionary computing techniques. The experimental results show that this algorithm finds more than one possible solution for a given source and destination and this makes it easy to find the next shortest path which exists other than the optimal solution.

As future works, the flexibility of selecting a town can be increased by assigning a fitness value for the routes and town nodes, which are visited more frequently. Then in the next generation the algorithm gives more priority to the cities and routes, which have a higher fitness. The technique is very similar to Ant Colony Optimization (ACO). Also assigning a penalty value to the fitness function according to the physical road conditions and the Traffic congestions in roads, gives the travelers to choose a suitable route to travel according road conditions and traffic congestion at a given time.

ACKNOWLEDGMENT

The First Author wishes to thank the department lecturers who guided in this work as well as his colleagues for keep the things up at all. Last but not least, this is a felicitation to loving parents for their guidance and encouragement throughout the life.

REFERENCES

[1] Peter G. Anderson, *Introduction to Genetic Algorithms,* Rochester, New York, 2002

[2] Sam Hsiung, James Matthews, “An Introduction to Genetic Algorithms,” http://www.generation5.org

[3] Gareth Jones, “Genetic and Evolutionary algorithms,” University of Sheffield, United Kingdom

[4] Daniel Ashlock, “Graph based genetic algorithms,” unpublished.