International Journal of Advance Research, IJOAR .org
Volume 1, Issue 4, April 2013, ISSN 2320-9194

APPLICATION OF GENETIC ALGORITHM TO SOLVE TRAVELING SALESMAN PROBLEM

Oloruntoyin Sefiu Taiwo, Olukehinde Olutosin Mayowa & Kolapo Bukola Ruka
Department of Computer Science & Engineering
Ladoke Akintola University of Technology, Ogbomoso
E-mail: oloruntoyin86@gmail.com

ABSTRACT

This research investigated the application of Genetic Algorithm capable of solving the traveling salesman problem (TSP). Genetic Algorithm are able to generate successively shorter feasible tours by using information accumulated in the form of a pheromone trail deposited on the edges of the TSP graph. Computer Simulations demonstrate that the Genetic Algorithm is capable of generating good solutions to both symmetric and asymmetric instances of the TSP. The method is an example, like simulated annealing, neural networks, and evolutionary computation of the successful use of a natural metaphor to design an optimization algorithm. A study of the genetic algorithm explains its performance and shows that it may be seen as a parallel variation of tabu search, with an implicit memory. Genetic algorithm is the most efficient in computational time but least efficient in memory consumption. The Genetic algorithm differs from the nearest neighbourhood heuristic in that it considers the nearest route while the neighbourhood heuristic considers the nearest path. The Genetic algorithm requires a system with parallel architecture for its optimal implementation. The activities of each genetic algorithm should be run as a separate operating system process.

Keywords: *Travelling Salesman Problem, Genetic Algorithms, Simulated Annealing.*

INTRODUCTION

The traveling salesman problem (TSP) is a well-known and important combinatorial optimization problem. The goal is to find the shortest tour that visits each city in a given list exactly once and then returns to the starting city. In contrast to its simple definition, solving the TSP is difficult since it is a Negative-Positive (NP) complete problem. Apart from its theoretical approach, the TSP has many applications. Some typical applications of TSP include vehicle routing, computer wiring, cutting wallpaper and job sequencing. The main application in statistics is combinatorial data analysis, e.g., reordering rows and columns of data matrices or identifying clusters. The NP-completeness of the TSP already makes it more time efficient for small-to-medium size TSP instances to rely on heuristics in case a good but not necessarily optimal solution is sufficient.

In this Research Work, genetic algorithm is used to solve Travelling Salesman Problem. Genetic algorithm is a technique used for estimating computer models based on methods adapted from the field of genetics in biology. To use this technique, one encodes possible model behaviors into "genes". After each generation, the current models are rated and allowed to mate and breed based on their fitness. In the process of mating, the genes are exchanged, crossovers and mutations can occur. The current population is discarded and its offspring forms the next generation. Also, Genetic Algorithm describes a variety of modeling and optimization techniques. Typically, the object being modeled is represented in a fashion that is easy to modify automatically. Then a large number of candidate models are generated and tested against the current data. Each model is scored and the "best" models are retained for the next generation. These models are then randomly perturbed (as in asexual reproduction) and the process is repeated until it converges. If the model is constructed so that they have "genes," the winners can "mate" to produce the next generation.

The Traveling Salesman problem (TSP) is one of the benchmark and old problems in Computer Science and Operations Research. The goal is to find the shortest tour that visits each city in a given list exactly once and then returns to the starting city.

Genetic algorithm is a technique used for estimating computer models based on methods adapted from the field of genetics in biology. To use this technique, one encodes possible model behaviors into "genes". After each generation, the current models are rated and allowed to mate and breed based on their

fitness. In the process of mating, the genes are exchanged, crossovers and mutations can occur. The current population is discarded and its offspring forms the next generation.
In a way of using GA in solving TSP, The following methods are used;

- *Simulated Annealing*, based on natural annealing processes.
- *Artificial Neural Networks*, based on processes in central nervous systems.
- *Evolutionary Computation* based on biological evolution processes.

The algorithms inspired by Evolutionary Computation are called *evolutionary algorithms*. These evolutionary algorithms may be divided into the following branches: *genetic algorithms* (Holland 1975), *evolutionary programming* (Fogel 1962), *evolution strategies* (Bremermann et al. 1965), *classifier systems* (Holland 1975), *genetic programming* (Koza 1992) and other optimization algorithms based on Darwin's evolution theory of natural selection and "survival of the fittest". In this project, we will only examine one of the above mentioned types of algorithms: genetic algorithms, although some of the exposed mutation operators have been developed in relation to evolutionary programming.

We consider these algorithms in combination with the *Travelling Salesman Problem* (TSP). The TSP objective is to find the shortest route for a travelling salesman who, starting from his home city has to visit every city on a given list precisely once and then return to his home city.
The main difficulty of this problem is the immense number of possible tours: (n-2)! /2 for *n* cities. Evolutionary algorithms are probabilistic search algorithms which simulate natural evolution.

Holland (1975) introduced *genetic algorithms*. In these algorithms the search space of a problem is represented as a collection of *individuals*. These individuals are represented by character strings which are often referred to as *chromosomes*. The purpose of using genetic algorithm to find the individual from the search space with the best "genetic material". The quality of an individual is measured with an evaluation function.

STATEMENT OF PROBLEM

In nature, there exist many processes which seek a stable state. These processes can be seen as natural optimization processes. Over the last 30 years, several attempts have been made to develop global optimization algorithms which simulate these natural optimization processes.

There are mainly three reasons why TSP has been attracted the attention of many researchers and remains an active research area. First, a large number of real-world problems can be modeled by TSP. Second, it was proved to be NP-Complete problem. Third, NP-Complete problems are intractable in the sense that no one has found any really efficient way of solving them for large problem size. Also, NP-complete problems are known to be more or less equivalent to each other; if one knew how to solve one of

them one could solve the rest.

The TSP finds application in a variety of situations such as automatic drilling of printed circuit boards and threading of scan cells in a testable VLSI circuit, X-ray crystallography, etc. The methods that provide the exact optimal solution to the problem are called **Exact Methods**.

An implicit way of solving the TSP is simply to list all the feasible solutions, evaluate their objective function values and pick out the best. However it is obvious that this "exhaustive search" is grossly inefficient and impracticable because of vast number of possible solutions to the TSP even for problem of moderate size. Since practical applications require solving larger problems, hence emphasis has shifted from the aim of finding exactly optimal solutions to TSP, to the aim of getting heuristically, 'good solutions' in reasonable time and 'establishing the degree of goodness'. Genetic algorithm (GA) is one of the best algorithms that have been used widely to solve the TSP instances.

1977 saw M. Grotschel's thesis which contained much of the polyhedral work on the TSP that he carried out jointly with M. Padberg. It also provided the solution for a 120-city instance by means of a cutting-plane algorithm, where cuts (sub tour inequalities and comb inequalities) were detected and added by hand to the linear programming relaxation. From that time onward, further work in cutting planes methods was carried out by the likes of: Miliotis, "Using cutting planes to solve the symmetric travelling salesman problem", Mathematical programming 15, 177 – 188, and M.W. Padberg and S. Hong, "On the symmetric Travelling salesman problem: a computational study", mathematical programming study 12, 78 – 107. As recently as 1991 M. Grotschel collaborated with O. Holland on, "Solution of large- scale symmetric travelling salesman problems".

The travelling salesman problem (TSP) is one which has commanded much attention of mathematicians and computer scientists specifically because it is so easy to describe and so difficult to solve.

The travelling salesman problem (TSP) is a deceptively simple combinatorial problem. It can be stated very simply:

> *A salesman spends his time visiting N cities (or nodes) cyclically. In one tour he visits each city just once, and finishes up where he started. In what order should he visit them to minimize the distance traveled?*

"Given a collection of cities, the travelling salesman must determine the shortest route that will enable him to visit each city precisely once and then return back to his starting point."

The travelling salesman problem can be re-phrased in term of a graph. Let a graph G=(X, A) whose vertices corresponds to the towns in the salesman's district and whose arc correspond to the roads

joining two towns. Also let the length d(x, y) ≥0 of the arc (x, y) equals the distance of the corresponding joining along arc (x, y). A cycles that includes each vertex in G at least once is called a salesman cycle. Salesman problem can be formulated on either undirected or directed graph.

THE VARIOUS METHODS AND ALGORITHM FOR T.S.P

These are some of the various methods for solving travelling salesman problem developed by mathematicians and computer scientist.

- **The shortest path algorithm**: This approximately finds the shortest path all pairs of node or vertices on a weighted graph.
- **The simple Insertion Algorithms**: This is based on the branch- and -bound algorithm. It derives an n-city tour from an (n-1) city tour. This process is applied recursively to build up to modify and produce a complete n-city solution.
- **Genetic Algorithm**: The idea of genetic algorithm is to stimulate the way nature uses evolution to solve T.S.P
- **The Elastic Net Methods**: This is a kind of artificial neural network, which is used primarily for optimization problem.
- **The Simulated Annealing Algorithm**; This involves the simulation of a physical process of annealing which is then used to develop a software program to solve T.S.P
- **Ant Algorithm**: This tries to use real ant abilities to solve various optimization problems.

Shortest path Algorithms

The shortest- paths problem involves a weighted, possibly directed graph described by the set of edges and vertices { E, V } Given a source vertex, s, the goal is to find the shortest existing path between s and any of the other vertices in the graph. Each path, therefore, will have the minimum possible sum of its component edges' (u, v) weights (w [u, v]). The method described here is **Dijkstra's algorithm**. Syswerda, G. (1991)

Dijkstra's Algorithm

The basic reasoning behind the algorithm is as follows:

- The algorithm gradually creates a tree from the vertices and some of the edges in the graph; the

root of this tree is the source of the search, vertex s, as specified at the outset of the algorithm. Thus for each vertex u, we maintain a **parent pointer** p[u] and a **distance** field d[u] (the distance, of course, is measured from the source.)

- The algorithm maintains a queue in which it keeps the vertices it has not yet examined. However, this is a priority queue, not a simple first – in – first – out queue. This priority queue is sorted using the d[] field of the vertices.

- Dijkstra's algorithm maintains a **solution set** S, which contains all the vertices, examined so far *whose minimal distance to the source has been determined*. As the algorithm progresses and examines more and more vertices, this set grows.

- At the outset, we set all parent pointers to NIL and all distance fields to infinity, except that of the source which we set to zero. Likewise, we set the solution set to empty.

- Each vertex in the priority queue is extracted in sequence (remember, the priority queue is ordered based on the d [] field of the vertices).

GENETIC ALGORITHMS

Genetic Algorithms (GAs) are adaptive heuristic search algorithm premised on the evolutionary ideas of natural selection and genetic. The basic concept of GAs is designed to simulate processes in natural system necessary for evolution, specifically those that follow the principles first laid down by Charles Darwin of survival of the fittest. As such they represent an intelligent exploitation of a random search within a defined search space to solve a problem.

First pioneered by John Holland in the 60s, Genetic Algorithms has been widely studied, experimented and applied in many fields in engineering worlds. Not only does GAs provide alternative methods to solving problem, it consistently outperforms other traditional methods in most of the problems link. Many of the real world problems involved finding optimal parameters, which might prove difficult for traditional methods but ideal for GAs. However, because of its outstanding performance in optimization, GAs have been wrongly regarded as a function optimizer. In fact, there are many ways to view genetic algorithms. Perhaps most users come to GAs looking for a problem solver, but this is a restrictive view [De Jong, 1993].

GAs were introduced as a computational analogy of adaptive systems. They are modelled loosely on the principles of the evolution via natural selection, employing a population of individuals that undergo selection in the presence of variation-inducing operators such as mutation and recombination (crossover). A fitness function is used to evaluate individuals, and reproductive success varies with fitness.

Genetic algorithms are algorithms developed from the concept of genetics. In genetics, it is understood that very living being's structure was developed from a 'data bank' of information (its gene, or genetic code) that controls the being's form and type. For example, a human being's gene contains his genetic information, which includes his skin colour, his eye hue, his teeth structure and even his brain capacity. The gene is so minute, it takes a very powerful microscope to view it, and its 'coding' is to complex; it took quite a while- and a whole brood of scientists years to decode it.

Control parameters

These are the parameters that govern the GA search process. Some of them are:

(a) **Population size**: - It determines how many chromosomes and thereafter, how much genetic material is available for use during the search. If there is too little, the search has no chance to adequately cover the space. If there is too much, the GA wastes time evaluating chromosomes.

(b) **Crossover probability**: - It specifies the probability of crossover occurring between two chromosomes.

(c) **Mutation probability**: - It specifies the probability of doing bit-wise mutation.

(d) **Termination criteria**: - It specifies when to terminate the genetic search.

Structure of genetic algorithms

GAs may be summarized as follows:

```
GA ( )
{ Initialize random population;
  Evaluate the population;
        Generation = 0;
            While termination criterion is not satisfied
            {Generation = Generation + 1;
              Select good chromosomes by reproduction procedure;
               Perform crossover with probability of crossover (Pc);
                   Select fitter chromosomes by survivor selection procedure;
                     Perform mutation with probability of mutation (Pm);
                         Evaluate the population;
                        }
                         }
```

The algorithm consists of the following fundamental steps

Initialization: Chromosomes are randomly created. At this point it is very important that the population is diverse otherwise the algorithm may not produce good solutions.

Evaluation: Each chromosome is rated how well the chromosome solves the problem at hand. A fitness

value is assigned to each chromosome.

Selection: Fittest chromosomes are selected for propagation into the future generation based on how fit they are.

Recombination: Individual chromosomes and pairs of chromosomes are recombined and modified and then put back in the population.

Methods through which Travelling salesman problem can be solved include simulated annealing, ant colony, Genetic algorithm to mention but few.

RESEARCH METHODOLOGY

We have developed a solution to the Traveling Salesman Problem **(TSP)** using a Genetic Algorithm **(GA)**. In the Traveling Salesman Problem, the goal is to find the shortest distance between *N* different cities. The path that the salesman takes is called a **tour**.
Testing every possibility for an *N* city tour would be *N!* math additions. 50 cities tour would have to measure the total distance of be 3.041 X 10^{64} different tours. Adding one more city would cause the time to increase by a factor of 51. Obviously, this is an impossible solution.
A genetic algorithm can be used to find a solution in less time. Although it might not find the best solution, it can find a near perfect solution for a 50 city tour in less than a minute. There are a couple of basic steps to solving the traveling salesman problem using a GA.

- First, create a group of many random tours in what is called a **population**. This algorithm uses a greedy initial population that gives preference to linking cities that are close to each other.
- Second, pick 2 of the better (shorter) tours **parents** in the population and combine them to make 2 new **child** tours. Hopefully, these children tour will be better than either parent.

A small percentage of the time the child tours are **mutated**. This is done to prevent all tours in the population from looking identical.

The new child tours are inserted into the population replacing two of the longer tours. The size of the population remains the same.

New children tours are repeatedly created until the desired goal is reached.

As the name implies, Genetic Algorithms mimic nature and evolution using the principles of **Survival of the Fittest**.

The two complex issues with using a Genetic Algorithm to solve the Traveling Salesman Problem are the **Encoding of the Tour** and **The Crossover Algorithm** that is used to combine the two parent tours to make the child tours.

In a standard Genetic Algorithm, the encoding is a simple sequence of numbers and Crossover is performed by picking a random point in the parent's sequences and switching every number in the sequence after that point. In this example, the crossover point is between the 3rd and 4th item in the list. To create the children, every item in the parent's sequence after the crossover point is swapped.

Parent 1	FAB	ECGD
Parent 2	DEA	CGBF
Child 1	FAB	CGBF
Child 2	DEA	ECGD

The difficulty with the Traveling Salesman Problem is that every city can only be used once in a tour. Let the letters in the above example represent cities, this child tours created by this crossover operation would be invalid if Child 1 goes to city F & B twice and never goes to cities D or E.

The encoding cannot simply be the list of cities in the order they are traveled. Other encoding methods have been created that solve the crossover problem. Although these methods will not create invalid tours, they do not take into account the fact that the tour "A B C D E F G" is the same as "G F E D C B A". To solve the problem properly the crossover algorithm will have to get much more complicated. Our solution stores the **links** in both directions for each tour. In the above tour example, Parent 1 would be stored as

City	First Connection	Second Connection
A	**F**	**B**
B	**A**	**E**
C	**E**	**G**
D	**G**	**F**

E	**B**	**C**
F	**D**	**A**
G	**C**	**D**

The crossover operation is more complicated than combining 2 strings. The crossover will take every link that exists in both parents and place those links in both children. Then, for Child 1 it alternates between taking links that appear in Parent 2 and then Parent 1. For Child 2, it alternates between Parent 2 and Parent 1 taking a different set of links.
For either child, there is a chance that a link could create an invalid tour where instead of a single path in the tour there are several disconnected paths. These links must be rejected. To fill in the remaining missing links, cities are chosen at random. Since the crossover is not completely random, this is considered a greedy crossover.

Eventually, this GA would make every solution look identical. This is not ideal. Once every tour in the population is identical, the GA will not be able to find a better solution. There are two ways around this. The first is to use a very large initial population so that it takes the GA longer to make all of the solutions the same. The second method is **mutation**, where some child tours are randomly altered to produce a new unique tour. This Genetic Algorithm also uses a greedy initial population. The city links in the initial tours are not completely random. The GA will prefer to make links between cities that are close to each other. This is not done 100% of the time, because that would cause every tour in the initial population to be very similar.

These parameters are to control the operation of the Genetic Algorithm:

- **Population Size** - The population size is the initial number of random tours that are created when the algorithm starts. A large population takes longer to find a result. A smaller population increases the chance that every tour in the population will eventually look the same. This increases the chance that the best solution will not be found.

- **Neighborhood / Group Size** -In each generation, this number of tours are randomly chosen from the population. The best 2 tours are the parents. The worst 2 tours get replaced by the children. For group size, a high number will increase the likelihood that the really good tours will be selected as parents, but it will also cause many tours to never be used as parents. A large group size will cause the algorithm to run faster, but it might not find the best solution.

- **Mutation %** - The percentage that each child after crossover will undergo **mutation** when a tour

is mutated; one of the cities is randomly moved from one point in the tour to another.

- **Nearby Cities** - As part of a greedy initial population, the GA will prefer to link cities that are close to each other to make the initial tours. When creating the initial population this is the number of cities that are considered to be closed.

- **Nearby City Odds %** - This is the percent chance that any one link in a random tour in the initial population will prefer to use a nearby city instead of a completely random city. If the GA chooses to use a nearby city, then there is an equally random chance that it will be any one of the cities from the previous parameter.

- **Maximum Generations** - How many crossovers are run before the algorithm is terminated

- **Random Seed** - This is the seed for the random number generator. By having a fixed instead of a random seed, you can duplicate previous results as long as all other parameters are the same. This is very helpful when looking for errors in the algorithm.

- **City List** - The downloadable version allows you to import city lists from XML files. Again, when debugging problems it is useful to be able to run the algorithm with the same exact parameters.

The starting parameter values are:

Parameter	Initial Value
Population Size	**10,000**
Group Size	**5**
Mutation	**3 %**
# Nearby Cities	**5**
Nearby City Odds	**90 %**

RESULTS AND ANALYSIS

MODELLING THE GENETIC ALGORITHMS

This section explains the design of genetic algorithm in order to imitate the collective behaviour of the TSP.

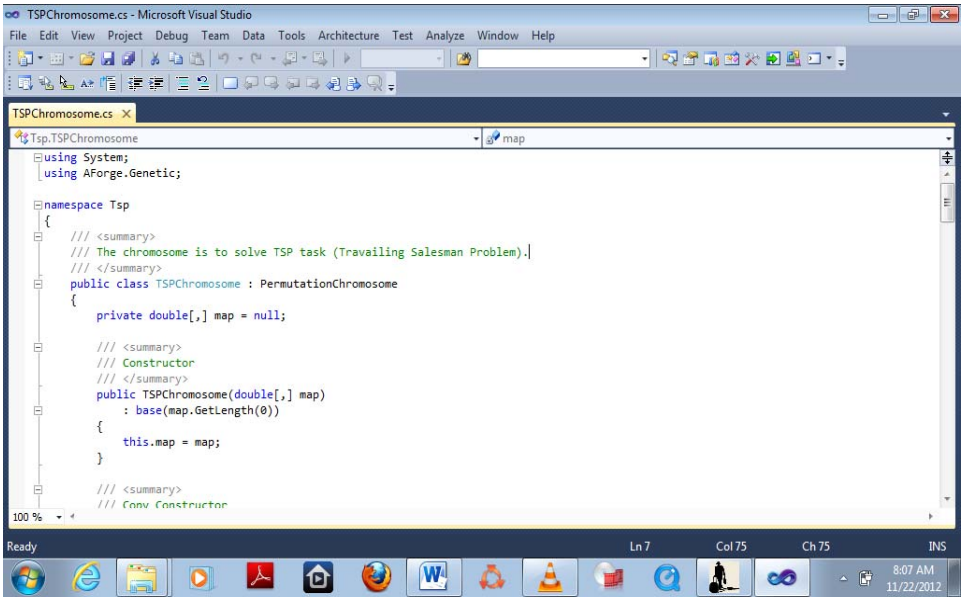
INTERFACE

The genetic algorithm was implemented using Visual C# Programming language. The choice of C# is as a result of the following;

- Object oriented technology
- Closeness to natural language
- Its good support for relational database
- Its execution speed and it's generation of true native execution i.e. it does not require additional files for its implementation.
- Good support of rich control structures and recursion.

The interface is designed by using the C# environment. All the elements of the environment were modelled using C#. Since these elements are simple, the implementation becomes simple. Many properties and methods of C# are specific for the interface.

the interface include



Figrure 4.1 program interface

Graph Name: This is name of each graph represented. It contains nodes and edges.

Graph Description: This shows how each node (each city) in the graph are connected to each other i.e. which loaction is linked with another as show in the screen shot of the interface. This also shows the way or manner in which the locations are visited.

Node Visited: This show the node that is visited at least once.

ENVIRONMENT

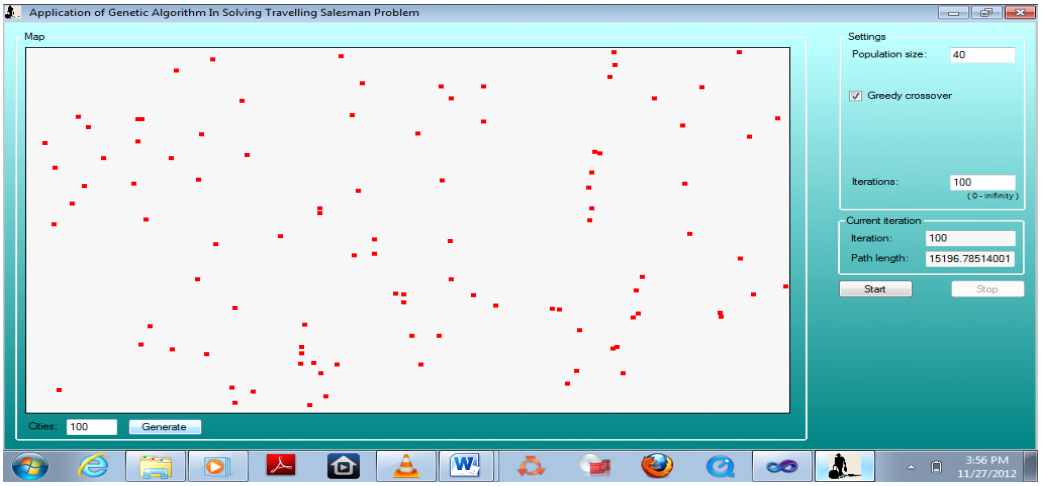


Figure 4.3 interface with generated cities/nodes

EXPLANATION

Critically looking at the environment, some parameters used need to be well understood in determining a successful and accurate tour.

CITIES: the number of cities specified will determine the number of node to be generated on the graph. The minimum and maximum number of cities that can be accomodated by this application are 5 and 200 respectively.

POPULATION SIZE: this specifies the maximum number of parent gene that can be combined in the process of mutation. The maximum number it can accommodate is 100.

GREEDY CROSS-OVER: this feature see to the mass selection of parent gene out the population size. It takes more parent to work upon when the feature is activated.

ITERATION: This is the number of time it uses to compute the graph.

PATH LENGTH: The is the total distance between each node/cities.

After the number of cities is known and written into the column provided, the "generate" key will be press to generate the cities using nodes on the screen. Then the number of performance needed in the computation of the tour i.e iteration is also written in the slot provided. The population size is also included after which it should be decided whether to mark of unmark the the greedy-crossover column.

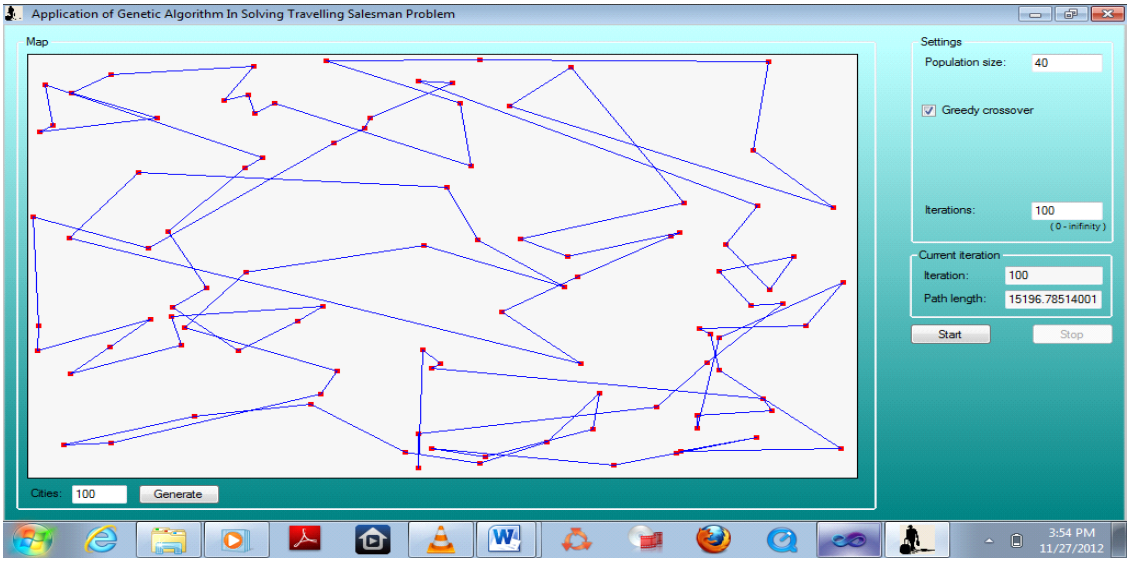


Figure 4.4 interface for the plotted cities generated

Considering the graph above and the foraging behaviour of genetic algorithm, it is easy to identify the

similarity between these two problems. While genetic algorithm try to find the best and shortest route between two places within an environment, a graph search algorithm tries to find the shortest path connecting two node within a graph. The main idea of the system proposed in this project is simply to put genetic algorithm on a graph to observe the accuracy with time.

A graph's node is taken at different places where genetic algorithm could stop during a travel and we call them cities. The edges of the graph will represent the route connecting cities. This virtual environment will be populated by individual genetic algorithm.

RESULTS AND FINDING

- The optimal number of genetic algorithm to be dispersed for foraging is n! where n is the number of nodes in the network.

- Genetic algorithm is the most efficient in computational time but least efficient in memory consumption.

- The genetic algorithm differs from the nearest neighbourhood heuristic in that it considers the nearest route while the nearest neighbourhood heuristic considers the nearest path.

- The genetic algorithm requires a system with parallel architecture for its optimal implementation.

CONCLUSION

The Genetic Algorithm is a very efficient and accurate optimization approach. Unfortunately it requires a large amount of memory and parallel system architecture for its complete implementation. The genetic algorithm was found to still be the best in solving combinatorial optimization problems (as it is general agreed by researcher around the world). We have successfully drawn out a model for the genetic algorithms, the design and the implementation.

It can be used successfully in large industries like Coca-Cola, Dangote Flour Mill to solve distribution problems, electronic industries like Intel® for circuit board drilling, order picking in a warehouse e.tc.

RECOMMENDATION

Genetic algorithm and its application certainly have a prospectus for further research for its successful implementation onto a number of problems including real world and industrial applications. Although the result produce by this algorithm are competitive compare with any other meta-heuristic methods, it is observed that the performance of the algorithm can be improved by coupling this with other method like Ant Algorithm, neural network, Simulated Annealing e.tc. The research work done by integrating with other techniques is very less and there is a lot potential to explore in this area.

REFERENCE

[1] B. Freisieben and P. Merz, New Genetic Local Search Operator for Travelling salesman Problem, Conference on Parallel Problem Solving From nature, app. 890-899, 1996.

[2] P. van Laarhoven and E. H. L. Aarts, Simulated Annealing: Theory and Applications, Kluwer Academic, 1987,

[3] M. Dorigo and L. M. Gambradella, Ant Colony System: A Cooperative Learning Approach to the Travelling Salesman Problem, JEEE Transaction on Evolutionary Computation, Vol. l, pp. 53-66, 1997.

[4] D.E. Goldberg, Genetic Algorithm in Search, Optimization and Machine Learning. Addison-Wesley, 1989.

[5] Naef Taher Al Rahedi and Jalal Atoum, Solving TSP problem using New Operator in Genetic Algorithms, American Journal of Applied Sciences 6(8):1586-1590, 2009.

[6] Goldberg D., Computer-Aided Pipeline Operations Using Genetic Algorithms and Rule Learning. Part I; Genetic Algorithms in pipeline Optimization, Engineering with Computer 3, 1987.

[7] Greffenstette, J., Genetic Algorithms Made Easy, 1991.

[8] Poli, R.et. al. (Eds), Evolutionary Image Analysis, Signal Processing and Telecommunications Springer, 1999.

[9] Seniw, D., A Genetic algorithm for the Travelling Salesman Problem, MSc Thesis, University of North Carolina, at Charlotte. http://www.heatonresearch.com/articales/65/page1.html. 1996.

[10] B. Freisleben and P. Merz, "A Genetic local search Algorithm for Solving Symmetric and Asymmetric Travelling Salesman Problems," International Conference On Evolutionary Computation, pp. 616-621, 1996.

[11] K. E. Nygard and C. H. Yang, "Genetic Algorithm for the Travelling Salesman Problem With Time Window", in Computer Science and Operations Research: New Development in their Interface, O. Balci, R. Sharda and S. A. Zenios Eds, Pergamon Press, pp. 411-423.

[12] H. Braun, "On Solving Travelling Salesman Problem by Genetic Algorithm", in Parallel Problem- Solving from Nature, Lecture Notes in Computer Science 496, H.P. Schwefel and R. Manner Eds, Springer-Verlag, app. 129-133.

[13] Ackley, D. H. (1987). *A Connectionist Machine for Genetic Hill climbing*. Kluwer Academic Publishers.

[14] Ambati, B. K., Ambati, J. &Mokhtar, M.M. (1991). Heuristic Combinatorial Optimization bySimulated Darwinian Evolution: A Polynomial Time Algorithm for the Traveling Salesman Problem. *Biological Cybernetics* **65**: 31–35.

[15] Banzhaf, W. (1990). The "Molecular" Traveling Salesman. *Biological Cybernetics* **64**: 7– 14.

[16] Beyer, H. G. (1992). Some Aspects of the 'Evolution Strategy' for Solving TSP Like Optimization Problems Appearing at the Design Studies of the 0.5 *TeVe+e*□□Linear Collider.

[17] In M¨anner, R. & Manderick, B. (eds.) *Parallel Problem Solving from Nature 2*, 361–370. Amsterdam: North Holland.

[18] Brady, R.M. (1985). Optimization Strategies Gleaned from Biological Evolution. *Nature* **317**: 804–806.

[19] Bremermann, H. J., Rogson, M. & Salaff, S. (1965). Search by Evolution. In Max field, M., CallahanA & Fogel, L. J. (eds.) *Biophysics and Cybernetic Systems*, 157–167.Washington: Spartan Books.

[20] Davis, L. (1985). Applying Adaptive Algorithms to Epistatic Domains. *Proceedings of the International Joint Conference on Artificial Intelligence*, 162–164.

[21] Davis, L. (ed.) (1991). *Handbook of Genetic Algorithms*. New York: Van Nostrand Reinhold.

[22] Fogel, L. J. (1962). Autonomous Automata. *Ind. Res.* **4**: 14–19.

[23] Fogel, D. B. (1988). An Evolutionary Approach to the Traveling Salesman Problem. *Biological Cybernetics* **60**: 139–144.

[24] Fogel, D. B. (1990).A Parallel Processing Approach to a Multiple Traveling Salesman Problem Using Evolutionary Programming. In Canter, L. (ed.) *Proceedings on the Fourth Annual Parallel Processing Symposium*, 318–326. Fullterton, CA.

[25] Fogel, D. B. (1993). Applying Evolutionary Programming to Selected Traveling Salesman Problems. *Cybernetics and Systems* **24**: 27–36.

[26] Fox, M. S. & McMahon, M. B. (1987). Genetic Operators for Sequencing Problems. In Rawlings, G. (ed.) *Foundations of Genetic Algorithms: First Workshop on the Foundations of Genetic Algorithms and Classifier Systems*, 284–300. Los Altos, CA: Morgan Kaufmann Publishers.

[27] Gunnels, J., Cull, P. & Holloway, J. L. (1994). Genetic Algorithms and Simulated Annealing for Gene Mapping. In Grefenstette, J. J. (ed.) *Proceedings of the First IEEE Conference on Evolutionary Computation*, 385–390. Florida: IEEE.

[28] Goldberg, D. E. & Lingle, Jr., R. (1985). Alleles, Loci and the TSP. In Grefenstette, J. J. (ed.) *Proceedings of the First International Conference on Genetic Algorithms and Their Applications*, 154–159. Hillsdale, New Jersey: Lawrence Erlbaum.

[29] Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading, MA: Addison Wesley.

[30]Gorges Schleuter, M. (1989). ASPARAGOS An Asynchronous Parallel Genetic Optimization Strategy. In Schaffer, J. (ed.) *Proceedings on the Third International Conference on Genetic Algorithms*, 422–427. Los Altos, CA: Morgan Kaufmann Publishers.

[31] Grefenstette, J., Gopal, R., Rosmaita, B. & Van Gucht, D. (1985). Genetic Algorithms for the TSP. In Grefenstette, J. J. (ed.) *Proceedings of the First International Conference on Genetic Algorithms and Their Applications*, 160–165. Hillsdale, New Jersey: Lawrence Erlbaum.

[32] Grefenstette, J. J. (ed.) (1987a). *Genetic Algorithms and Their Applications: Proceedings of the Second International Conference*. Hillsdale, New Jersey: Lawrence Erlbaum.

[33]Grefenstette, J. J. (1987b). Incorporating Problem Specific Knowledge into Genetic Algorithms.

[34] In Davis, L. (ed.) *Genetic Algorithms and Simulated Annealing*, 42–60. Los Altos, CA: Morgan Kaufmann.

[35] Holland, J. (1975). *Adaptation in Natural and Artificial Systems*. Ann Arbor: University of Michigan Press.

[36] Homaifar, A. & Guan, S. (1991). *A New Approach on the Traveling Salesman Problem by Genetic Algorithm*. Technical Report, North Carolina A&T State University.

[37] Homaifar, A., Guan, S. & Liepins, G. E. (1993). A New Approach on the Traveling Salesman Problem by Genetic Algorithms. In Forrest, S. (ed.) *Proceedings of the Fifth International Conference on Genetic Algorithms*, 460–466.

[38] Jog P., Suh, J. Y. & Van Gucht, D. (1989). The Effects of Population Size, Heuristic Crossover and Local Improvement on a Genetic Algorithm for the Traveling Salesman Problem. In Schaffer, J. (ed.) *Proceedings on the Third International Conference on Genetic Algorithms*, 110–115. Los Altos, CA: Morgan Kaufmann Publishers.

[39] Johnson, D. S. (1990). Local Optimization and the Traveling Salesman Problem. *Proc. 17th Colloq. Automata, Languages and Programming*. Springer Verlag.

[40] Kirkpatrick, S., Gelatt, C. D. & Vecchi, M. P. (1983). Optimization by Simulated Annealing. *Science* **220**: 671–680.

[41] Koza, J. R. (1992). *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press.

[42] Larra˜naga, P., Kuijpers, C. M. H., Poza, M. & Murga, R. H. (1996a) Decomposing Bayesian Networks: Triangulation of the Moral Graph with Genetic Algorithms. *Statistics and Computing* (to be published).

[43] Larra˜naga, P., Kuijpers, C. M. H., Murga, R. H. & Yurramendi, Y. (1996). Searching for the Best Ordering in the Structure Learning of Bayesian Networks. *IEEE Transactions on Systems, Man and Cybernetics* **26**(4): 487–493.

[44] Larra˜naga, P., Inza, I., Kuijpers, C. M. H., Gra˜na, M. & Lozano, J. A. (1996). Algorithms Gen´eticos en el Problema delViajante de Comercio. *Informatica y Automatica* (submitted).

[45] Lauritzen, S. L. & Spiegel halter, D. J. (1988). Local Computations with Probabilities on Graphic Structures and Their Application on Expert Systems. *Journal of the Royal Statistical Society, Series B* **50**(2): 157–224.

[46] Lawler, E. L., Lenstra, J. K., Rinnooy Kan, A. H. G. & Shmoys, D. B. (eds.) (1985). *The Travelling Salesman Problem: A Guided Tour of Combinatorial Optimization*. Chichester: Wiley.

[47] Lidd, M. L. (1991). *The Travelling Salesman Problem Domain Application of a Fundamentally New Approach to Utilizing Genetic Algorithms*. Technical Report, MITRE Corporation.

[48] Liepins, G. E., Hilliard, M. R., Palmer, M. & Morrow, M. (1987). Greedy Genetics. In Grefenstette, J. J. (ed.) *Genetic Algorithms and Their Applications: Proceedings of the Second International Conference*, 90–99. Hillsdale, New Jersey: Lawrence Erlbaum.

[49] Lin, S. (1965). Computer Solutions on the Travelling Salesman Problem. *Bell Systems Techn. J.* **44**: 2245–2269.

[50] Lin, S. & Kernighan, B. W. (1973). An Effective Heuristic Algorithm for the Traveling Salesman Problem. *Operations Research* **21**: 498–516.

[51] Lin, F.T., Kao, C.Y. & Hsu, C.C. (1993). Applying the Genetic Approach to Simulated Annealing in Solving NP Hard Problems. *IEEE Transactions on Systems, Man, and Cybernetics* **23**(6): 1752–1767.

[52] Lozano, J. A., Larra˜naga, P. & Gra˜na, M. (1996). Partitional Cluster Analysis with Genetic Algorithms: Searching for the Number of Clusters. *Fifth Conference of International Federation of Classification Societies*, 251–252. Kobe, Japan.

[53] Matthews, R. A. J. (1993). The Use of Genetic Algorithms in Cryptanalysis. *Cryptologia* **XVII** (2): 187–201.

[54] Michalewicz, Z. (1992). *Genetic Algorithms + Data Structures = Evolution Programs*. Berlin Heidelberg: Springer Verlag.

[55] M¨uhlenbein, H., Gorges Schleuter, M. & Kr¨amer, O. (1987). New Solutions to the Mapping Problem of Parallel Systems: The Evolution Approach. *Parallel Computing* **4**: 269–279.

[56] M¨uhlenbein, H., Gorges Schleuter, M. & Kr¨amer, O. (1988). Evolution Algorithms in Combinatorial Optimization. *Parallel Computing* **7**: 65–85.

[57] M¨uhlenbein, H. (1989). Parallel Genetic Algorithms, Population Genetics and Combinatorial Optimization. In Schaffer, J. (ed.) *Proceedings on the Third International Conference on Genetic Algorithms*, 416–421. Los Altos, CA: Morgan Kaufmann Publishers.

[58] M¨uhlenbein, H. & Kindermann, J. (1989). The Dynamics of Evolution and Learning – Towards Genetic Neural Networks. In Pfeiffer, J. (ed.) *Connectionism in Perspectives*.

[59] M¨uhlenbein, H. (1991). Evolution in Time and Space – The Parallel Genetic Algorithm. In Rawlins, G. (ed.) *Foundations of Genetic Algorithms*. Los Altos, CA: Morgan Kaufmann.

[60] Oliver, I. M., Smith, D. J. & Holland, J. R. C. (1987). A Study of Permutation Crossover Operators on the TSP. In Grefenstette, J. J. (ed.) *Genetic Algorithms and Their Applications: Proceedings of the Second International Conference*, 224–230. Hillsdale, New Jersey:

[61] Lawrence Erlbaum. Or, I.(1976). *Travelling Salesman Type Combinatorial Problems and Their Relation to the Logistics of Regional Blood Banking*. PhD Thesis, Northwestern University.

[62] Prinetto, P., Rebaudengo, M. & Sonza Reorda, M. (1993). Hybrid Genetic Algorithms for the Traveling Salesman Problem. In Albrecht, R. F., Reeves, C. R. & Steele, N. C. (eds.) *Artificial Neural Nets and Genetic Algorithms*, 559–566. Wien: Springer Verlag.

[63] Rechenberg, I. (1973). *Optimierung Technischer Systeme Nach Prinzipien der Biologischen Information*. Stuttgart: Frommann Verlag.

[64] Reinelt, G. (1991). TSPLIB – A Traveling Salesman Library. *ORSA Journal on Computing* **3**(4): 376–384.

[65] Schaffer, J. (ed.) (1989). *Proceedings on the Third International Conference on Genetic Algorithms*. Los Altos, CA: Morgan Kaufmann Publishers.

[66] Schwefel, H.P. (1975). *Evolutions strategie und Numerische Optimierung*. Doctoral Thesis Diss. D 83, TU Berlin.

[67] Seniw, D. (1991). *A Genetic Algorithm for the Traveling Salesman Problem*. MSc Thesis, University of North Carolina at Charlotte.

[68] Spillman, R., Janssen, M., Nelsonn B. & Kepner, M. (1993). Use of a Genetic Algorithm in the Cryptanalysis Simple Substitution Ciphers. *Cryptologia* **XVII** (1): 31–44. SPSSX, User's Guide (1988). 3rd Edition.

[69] Starkweather, T., McDaniel, S., Mathias, K., Whitley, C. &Whitley, D. (1991). A Comparison of Genetic Sequencing Operators. In Belew, R. & Booker, L. (eds.) *Proceedings on the Fourth International Conference on Genetic Algorithms*, 69–76. Los Altos, CA: Morgan Kaufmann Publishers.

[70] Suh, J. Y. & Van Gucht, D. (1987). Incorporating Heuristic Information into Genetic Search. In Grefenstette, J. J. (ed.) *Genetic Algorithms and Their Applications: Proceedings of the Second International Conference*, 100–107. Hillsdale, New Jersey: Lawrence Erlbaum.

[71] Syswerda, G. (1991). Schedule Optimization Using Genetic Algorithms. In Davis, L. (ed.) *Handbook of Genetic Algorithms*, 332–349. New York: Van Nostrand Reinhold.

[72] Ulder, N. L. J., Aarts, E. H. L., Bandelt, H.J., Van Laarhoven, P. J. M. & Pesch, E. (1990). Genetic Local Search Algorithms for the Traveling Salesman Problem. In *Parallel Problem Solving from Nature*, 106–116. Berlin Heidelberg: Springer Verlag.

[73] Whitley, D., Starkweather, T. & D'Ann Fuquay (1989). Scheduling Problems and Travelling Salesman: The Genetic Edge Recombination Operator. In Schaffer, J. (ed.) *Proceedings on the Third International Conference on Genetic Algorithms*, 133–140. Los Altos, CA: Morgan Kaufmann Publishers.

[74] Whitley, D., Starkweather, T. & Shaner, D. (1991). The Traveling Salesman and Sequence Scheduling: Quality Solutions Using Genetic Edge Recombination. In Davis, L. (ed.) *Handbook of Genetic Algorithms*, 350–372. New York: Van Nostrand Reinhold.

[75] Fozia hanif khan, nasiruddin khan, syed inayatullah, and shaikh tajuddin nizami: solving tsp problem by using genetic algorithm. *International Journal of Basic & Applied Sciences* IJBAS Vol: 9 No: 10

[76] P. LARRAN˜ AGA, C.M.H. KUIJPERS, R.H. MURGA, I. INZA and S. DIZDAREVIC (1999): Genetic Algorithms for the Travelling Salesman Problem: A Review of Representations and Operators. *Artificial Intelligence Review* **13:** 129–170, 1999 *Kluwer Academic Publishers*