Volume 2, Issue 2 (2014) 436-440 ISSN 2347 - 3258
International Journal of Advance Research and Innovation

Analysis of TSP Optimization Problems using Genetic Algorithm

Aparna *, **Waseem Ahmad**
Department of Computer Science & Engineering, AFSET, Faridabad, India

Article Info

Article history:
Received 5 January 2014
Received in revised form
20 February 2014
Accepted 28 February 2014
Available online 15 June 2014

Keywords

Genetic Algorithms (GA),
Mutation operator,
Crossover operator
TSPGA

Abstract

Initiation of this paper demarcates Genetic Algorithm, as it paradigm shift in soft computing to solve optimization problems. This paper perposed the application of GAs to TSP by examining combinations on different algorithms for the binary and unary operators used to generate better solutions and minimize the search space calls. Operational view of genetic Algorithm to solve TSP.

This paper describes the number of crossover operators and mutation operators that can be applied with path representation in order to solve TSP. The TSP is, given a collection of cities, the problem is to determine the shortest route which visits each city precisely once and then returns to its starting point.TSP is complex as it is a NP-complete problem. This literature proposes a heuristic approach for solving TSP with Genetic Algorithm as TSPGA. Genetic algorithm (GA's) has been used as search techniques on many NP (Non-Deterministic Polynomial Time) problems. Three binary and two unary operators were tested: To find shortest path that travels through every city in a provided set of cities exactly once and travels back to the initial city. It is a faster solution but not necessarily an optimal solution.

1. Introduction

Genetic Algorithm represents a paradigm shift in the soft computing to solve combinatorial optimization problems which is based on the principle of natural selection. The formal algorithm which is based on the Darwin's theory was developed by John Holland and his students in 1970.The continuous improvement in the price/performance value of GA's has made them useful for solving optimization method. The Selection operator selects two members of the present generation to be used in the next operations: crossover and mutation. The most important aspect of the GA are:

1. Definition of the objective function
2. Definition and implementation of the genetic representation.
3. Definition and implementation of the genetic operators. Once these three have been defined, the genetic algorithm should work fairly well.

2. Problem Definition

Path representation have been many different representations used to solve the TSP problem using genetic algorithms (GA) as:

- Binary representation,
- Matrix representation,
- Adjacency representation,
- Ordinal representation
- Path representation

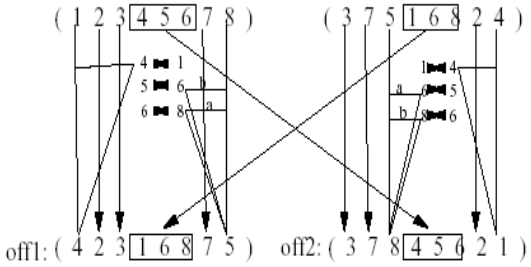
3. Operators

There are number of crossover operators and mutation operators that can be applied with path representation in order to solve TSP. A portion of one parent's string is mapped onto a portion of the other parents string and the remaining information is exchanged.

Corresponding Author,
E-mail address: tomaraparna030@gmail.com

Example

Two parent tours:
(12345678) and (37516824)



First, it selects uniformly at random two cut points along the strings, which represent the parent's tours. The sub strings between the cut points are called the mapping sections. In the above example they define the mapping 4-1, 5-6, and 6-8. Now the mapping section of the first parent is copied into the second offspring and the mapping section of the second parent is copied into the first offspring.

Iffspring1 (xxx168xx) and offspring2: (xxx456xx)

Then offspring 'i' ('i'=1, 2) is filled up by copying the elements of the I-th parents. In case a city is already present in the offspring it is replaced according to the mapping. For example the first element of offspring 1 would be a 1 like the first element of the first parent. However there is already a 1 present in offspring1. Hence, because of the mapping 1-4 we choose the first element of the offspring 1 to be a 4. The second, third and seventh elements of offspring1 can be taken from the first parent. However, the last element of the offspring 1 would be an 8, which is already present. Because of the mapping 6-6 and 6-5, it is chosen to be a 5. Hence

Offspring1 = (42316875) similarly
Offspring2 = (37846521)

3.1 Greedy Sub Tour Crossover

Example

Consider a tour g=(DHBACFGE).

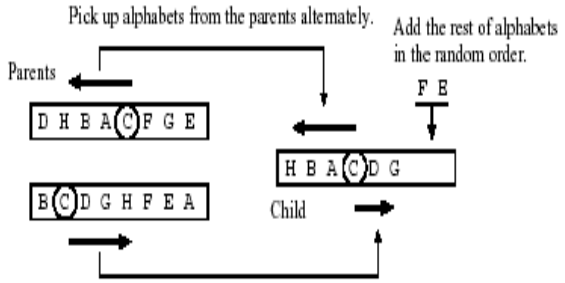


Fig: 2. Greedy Subtour Crossover

Suppose that chromosomes of parents are
ga = (DHBACFGE) and gb = (BCDGHFEA).

First, choose one town at random. In this example, town C is chosen. Then, x = 4 and y = 1 because a4 = C and b1 = C respectively. Now the child g is (C). Next, pick up towns from the parents alternately. Begin with a3 (town A) because x 4 _ 1 = 3, and next is b2 (town D) because y 1+ 1 = 2. The child becomes g = (A;C;D). In the same way, add a2 (town B), b3 (town G), a1 (town H), and the child becomes g = (HBACDG). Now the next town is b4 = H and town H has already appeared in the child (remember the salesperson may not visit the same town twice), so we can't add any more towns from parent gb. Therefore we add towns from parent ga. The next town is a0 = D, but D is already used. Thus we can't add towns from parent ga, either. Then, we add the rest of the towns, i.e., E and F, to the child in the random order. Finally the child is g = (HBACDGFE).

3.2 Greedy Exchange Mutation

In this method two cities are selected randomly and exchanged.

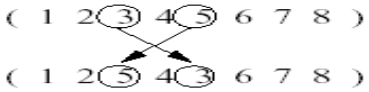
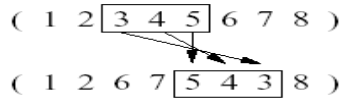


Fig: 2. Exchange Mutation

3.3 Inversion Random Displacement

In this method a sub tour is selected in random and it is removed from the tour and inserted it in a randomly selected position. However the sub tour is inserted in reverse order



The objective of TSP is to minimize the distance or path of the shortest length and the minimum cost on an undirected graph that represents cities or node to be visited. The successor of the last node in the permutation is the first one. The Euclidean distance d, between any two cities with coordinate (x1, y1) and (x2, y2) is calculated by:

$$d = \sqrt{(|X_1 - X_2|)2 + (|Y_1 - Y_2|)2} \quad (1)$$

4. Tsp Using GA

The work in this paper aims to develop a computer software system for solving the TSP based on Genetic Algorithm (TSPGA).

The number of cities involved in the search is entered and named as the letters {a, b, c…}, and then Euclidean distances between the cities are calculated or entered. An initial solution is generated as a sequence of letters. Then a fitness function is used and two solutions are selected leading to the emergence of a new born solution which replaces one of the parent solutions.

4.1 GA Components

The successive generations are derived by applying the selection, crossover and mutation operators to the previous tour population.

4.2 The Fitness Function

A fitness function evaluation is incorporated to assigns a value to each organism, noted as f_i. This f_i value is a figure of merit which is calculated by using any domain knowledge that applies. The method employed here was to calculate the total Euclidean distance D_i for each organism first, then compute f_i by using the following equation

$$f_i = D_{max} - D_i \quad (2)$$

Where D_{max} is the longest Euclidean distance over organisms in the population.

4.3 Selection Operations

The selection operator chooses two members of the Mutation present generation to participate in the next operations of crossover and mutation

4.4.1 Specialized Crossover Operator

Two points crossover operator would have to be modified to work with such problems. Exchanging parts of two solutions will usually produce an invalid solution. Below an example that shows two parents of length 9. Select two random points in parents; say 3 and 6, then the alleles between those points are swapped to generate two children solutions. Both of the resulting children solutions are invalid [18].

Table: 1. An Example For Path Fitness Function

Sr. No.	Orgnasim	D_i	f_i	P_i	S_i
1	BEFCAD	122	0	0.01	1
2	FEBACD	101	21	0.10	2
3	CDAFBE	80	42	0.19	2
4	DAFCBE	50	72	0.31	2
5	BDAEFC	30	92	0.40	3

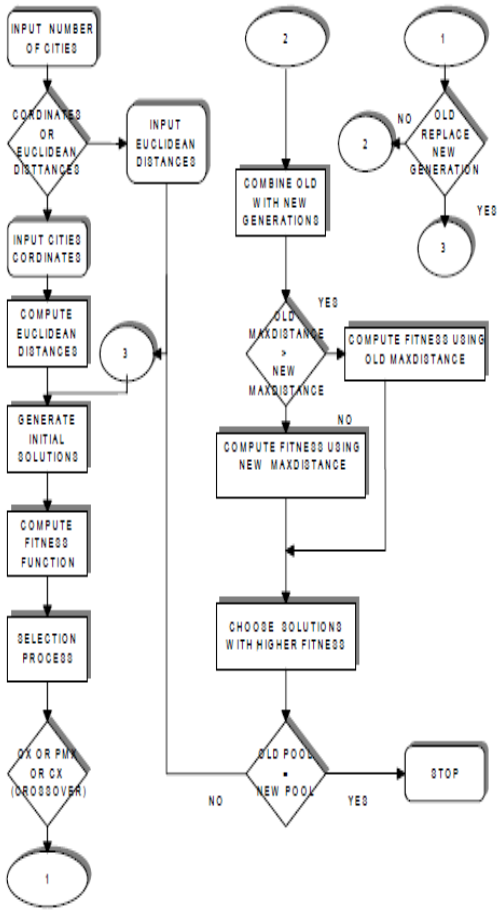


Fig: 3. The Flow Diagram For TSP Using GA Software System

Three types of special crossover operators reported for permutation problems are selected to be examined and used in the proposed TSPGA system.

They are:

Order crossover (OX), Partially matched crossover (PMX), and Cycle crossover (CX), as described briefly below.

A. Order Crossover (Ox)

To apply OX, two random cross points are selected. Alleles from parent1 that fall between the two cross points are copied into the same positions of the offspring. The remaining allele order is determined by parent2. Non-duplicative alleles are copied from parent2 to the offspring beginning at the position following the second cross point. Both the parent2 and the offspring are traversed circularly from that point. A copy of the parent′s next non-duplicative allele is placed in the next available child position [17]. An Example of OX is given below with two random cross points; 3 and 6, the alleles in the crossing sites from parent1 (GHB) are copied into the same positions of the child. The alleles after second cross point in parent2 (BA), B is skipped since it already exists in child, therefore only A is copied to child at position 7. Traverse parent2 circularly, the alleles (HDE), skip H to D which is copied to child at position 8. Traversed Child circularly, the alleles from parent2 (EF) are copied to child at positions 1 and 2. Finally, skip G to C and copy it to child at position 3.

Example

Parent1 = FBC|AHG|DE
Parent2 = ABC|DEF|GH
→ *Child* = DBC|AHG|FE

B. Partially Matched Crossover

PMX proceeds just as OX. Alleles from parent1 that fall between two randomly selected crossing sites are copied into the same positions of the offspring first, alleles in parent2 not within crossing sites are copied to the corresponding positions within the offspring. Next each allele of parent2 within the crossing sites is placed in the offspring at the position occupied in parent2 by the allele from parent1 that displaced it [17], see the example below. The random crossing points are 3 and 6, the alleles in the crossing site of parent1 (AHG) displace in the child (DEF), and then the alleles (BC) in parent2 which are not in crossing site are copied into the same positions of the child. D goes to position 1 which is the position with respect to parent2, displacing allele A. Finally E and F are placed in the positions of H and G, respectively.

Example

Parent1 = FBC|AHG|DE
Parent2 = ABC|DEF|GH
→ *Child* = DBC|AHG|FE

C. Cycle Crossover

CX performs recombination under the constraint that each allele value comes from one parent or the other [19]. CX does not use crossing sites, but a cycle is defined in a manner similar to an algebraic permutation group [17], see the example below, Comparing the strings of parent1 with parent2, F displaces A, A displaces D, D displaces G, and G displaces F. This forms the cycle FADG. The remaining alleles are filled from parent2 in the corresponding positions, BC, E, and H.

Example

Parent1 = FBCAHGDE
Parent2 = ABCDEFGH
→ *Child* = FBCAEFGDH

4.4.2 Specialized Mutation Operator (SMX)

The mutation operator is a permutation problem arbitrarily changing single allele value that does not preserve allele uniqueness. The method frequently used for permutation problems is to interchange two randomly selected positions, thus preserving allele uniqueness [17]. The example below uses one of the previous child solutions. Let the crossing points be 2 and 6 selected randomly. The alleles B and G are swapped positions in the mutated child Solution.

Example:

Old Child = F-BCAE-GDH → *Mutated Child* = FGCAEBDH

The system is designed to run in one of two options; default and customized as requested by the user. This is clearly shown in the flow diagram of Fig. 4.1

A. Default System

The user enters the required data to produce solutions only, but he/she has neither influence nor choice to control the process. The selection of the crossover, mutation operations and mating pool strategies are chosen based on the experimental results, rather than user choice.

It is executed according to the following steps.

1- The system prompts to enter number of cities.
2- The system prompts to enter coordinates of cities, then it computes Euclidean distances between cities or they may be entered directly depends on user desire.
3- After distance matrix becomes available, the system generates randomly initial solutions and Computes longest Euclidean distance.
4- Finally the GA system continues processing till a solution is reached.

B. Customized System

This option operates the system according to user choices. The system prompts to enter the required data as in default option, then prepares distance matrix.

The user has the selection of the followings:

1- The version of crossover operator, i.e. OX, PMX, CX.
2- The strategy of maintaining the mating pool can be one of the following:

Strategy1: the new generation replaces old generation.
Strategy2: the combination of old and new generations.

Choosing *strategy1* of the above have given noisy results and took long implementation time. Therefore, *strategy2* was adopted, which determines the longest (maximum) .The resulted solutions were sorted in descending order according to fitness value then the solution with higher fitness was chosen.

5. Results

GA as the name implies, this algorithm gets its idea from genetics.

Algorithm 1

Step 0. Obtain the maximum number of individuals in the population P and the maximum number of generations G from the user, generate P solutions for the first generation's population randomly, and represent each solution as a string. Set generation counter Ngen=1.

Step 1. Determine the fitness of each solution in the current generation's population and record the string that has the best fitness.

Step 2. Generate solutions for the next generation's population as follows:

1. Retain 0.1P of the solutions with the best fitness in the previous population.

2. Generate 0.89P solutions via mating, and

3. Select 0.01P solutions from the previous population randomly and mutate them.

Step 3. Update $N_{gen}= N_{gen+1}$. If Ngen ≤ G, go to Step 1. Otherwise stop.

In our project, we used the two-point crossover method, given two parent chromosomes $\{x_1,x_2,\ldots x_n\}$ and $\{y_1,y_2,\ldots,y_n\}$, two integers r and s are randomly selected, such that 1 £ r £ s £ n. And the genes in positions r to s of one parent are swapped with those of the other to get two off springs as follows.

$\{x_1, x_2,\ldots, x_{r-1}, y_r, y_{r+1},\ldots, y_s, x_{s+1}, x_{s+2},\ldots, x_n\}$
$\{y_1, y_2,\ldots, y_{r-1}, x_r, x_{r+1},\ldots, xs, y_{s+1}, y_{s+2},\ldots, y_n\}$

Either both or the better of the two offspring is then included in the new population. This mutation procedure is repeated until the required number of offspring is generated. To avoid generating infeasible solutions, we also used the partially matched crossover method. Set t=r, and swaps genes x_t and y_t if xt ≠ yt in the first parent $x=\{x_1, x_2\ldots x_{r-1}, x_r, x_{r+1},\ldots, xs, x_{s+1}, x_{s+2},\ldots, x_n\}$. This swapping is done for t=r through s, one by one, and then we have a feasible offspring. The same is done for $y=\{y_1, y_2,\ldots, y_{r-1}, yr, y_{r+1},\ldots, y_s, y_{s+1}, y_{s+2},\ldots, y_n\}$. Because the partially matched crossover method always swaps genes within a parent, feasible offspring are obtained. Therefore, our model uses mutation to generate a population by swapping any two randomly selected genes.

Algorithm 2

Step 1. Find the point jc which is closed to city i.

Step 2. Point j_c is moved with its neighbors towards city i to form a ring. The moved distance is measured by function f(G,n).G is the gain factor and n is the distance measured along the loop between nodes j and jc.

n=inf (j- jc (mod N), jc -j(mod N))

Point j is moved from current position to a new one.
The function f is defined to be

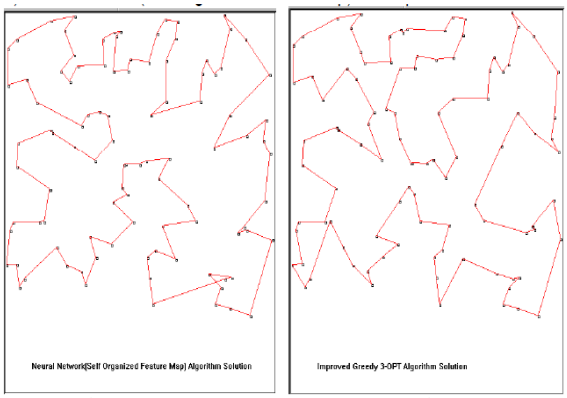
f(G,n)=sqrt(1/2)*exp(-n2/G2)

This means:

- When (G, n), all points move towards city I with the equal strength sqrt(1/2).

- When (G, n) only point jc travels towards city i.

Reducing the gain in the end of a complete survey is done by G←(1- α)G. The parameter α is required to be adjusted .The parameter α id related to the number of cities which are required to be monitored. The gain decreases from a high initial G_i which makes sure that large moves for all points in every iteration to a sufficiently small G_f for which the network is sustained. The newly made point is inserted as a neighbor to winner into the ring, and its coordinated are same in the plane.



Length : 4035.32
Time : 12 sec

Length : 43995.19

6. Conclusions and Remarks

The TSPGA system was developed to assist the user to obtain optimal tour to visit n cities. It has proposed the use

of advanced operators of Genetic Algorithms in order to enhance the rate of divergence, and achieved tours with reasonable time. It has used an interactive user interface enabling users to handle most system features. The developed selection algorithm used helps to select a suitable strategy for selecting a pair of parents for crossover operation. Besides, newly developed fitness function is adopted and has produced reasonable results. Furthermore, the use of man-machine interaction to guide the algorithmic procedure in a way different from that used in the proposed systems, as it allows the user to intervene at each step in the system executions to choose the relevant path or schedule from the current generation to produce a satisfactory results.

Reference

[1] D. L. Applegate, R. E. Bixby, V. Chvátal, W. J. Cook, The Traveling Salesman Problem: A Computational Study. Princeton University Press, 2006

[2] Schrijver, Alexander, On the history of combinatorial optimization, Handbook of Discrete Optimization (K. Aardal, G. L. Nemhauser, R. Weismantel, eds.), Elsevier, Amsterdam, 2009, 1-68

[3] E. L. Lawler, J. K. Lenstra, A. H. G Rinnooy Kan, D. B. Shmoys, The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization, Wiley address Interscience, Chichester, 1985

[4] P. Moscato, M. G. Norman, A `Memetic', Approach for the Traveling Salesman Problem. Implementation of a Computational Ecology for Combinatorial Optimization on Message-Passing a Systems, Parallel Computing and ransputer Applications, edited by M. Valero, E. Onate, M. Jane, J.L. Larriba and B. Suarez, Ed. IOS Press, Amsterdam, 1992, 187 –194.

[5] M. Lalena, Traveling Salesman Problem Using Genetic Algorithms. http://www.lalena.com/ai/tsp/, 2011

[6] D. Whitley, K. Mathias, Genetic Operators, the Fitness Landscape and the Traveling Salesman Problem, Parallel Problem Solving from Nature-PPSN 2. R. Manner and B. Mandrake North Holland-Elsevier, eds., 1992, 219 – 228.

[7] S. Kedar, Naphade, Dilek Tuzun, Initializing the Hopfield-Tank Network for the TSP using a convex hull: A Computational Study. Proceedings of the Artificial Neural Networks in Engineering (ANNIE'95) Conference, 1995, 399 – 404

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

[9] P. Larranaga, C. M. H. Kuijpers, R. H. Murga, I. Inza, S. Dizdarevic, Genetic Algorithms for the Travelling Salesman Problem: A Review of Representations and Operators, Artificial Intelligence Review. 13, 1999, 129-170.

[10] S. R. Shubhra, B. Sanghamitra, K. Pal. Sankar, New Operators of Genetic Algorithms for Traveling Salesman Problem, IEEE, 0-7695-2128-2/04, 2004.

[11] Ji. Ping, Ho. William, The Traveling Salesman and the Quadratic Assignment Problem: Integration, Modeling and Genetic Algorithm, Int. Symposium on OR and its Applications, 2005, 198-205

[12] C. Ding, Ye. Cheng, M. He, Two-Level Genetic Algorithm for Clustered Traveling Salesman Problem with Application in Large-Scale TSPs", Tsinghua Science and Technology, 12(4), 2007, 59-465

[13] P. Borovska, T. Ivanova, H. Salem, Efficient Parallel Computation of the Traveling Salesman Problem on Multicomputer Platform, Proceedings of the International Scientific Conference 'Computer Science' 2004, Sofia, Bulgaria, 2004, 74-79

[14] R. Gremlich, A. Hamfelt, H. de Pereda, V. Valkovsky, Genetic Algorithms with Oracle for the Traveling Salesman Problem", proceedings of world academy of science, Engineering and Technology, 2005, 1307-6884

[15] Q. Jiang, R. Sarker, H. Abbass, Tracking moving targets in the nonstationary travelling salesman problem," Complexity International, 11, 2005, 171-179

[16] V. Lawrence, A. Snyder, M. S. Daskin, A random-key genetic algorithm for the generalized traveling salesman problem, Available online since, 2005

[17] M. Bakhouya, J. Gaber, An Immune Inspired-based Optimization Algorithm: Application to the Traveling Salesman Problem, AMO, Advanced Modeling and Optimization, 9(1), 2007

[18] B. P. Buckles, P. E. Petry, R. I. Kuester, Schema Survival Rates and Heuristic Search in Genetic Algorithms, IEEE, 1990, 86-91

[19] D. T. Phillips, A. Rarindran, J. J. Solberg, Operations Research: Principles and Practice", John Wiely and Sons Inc, 1976

[20] D. E. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning, Addison Wesley Publishing Company Inc. 1989.

[21] Naef Taher Al Rahedi, Jalal Atoum, Solving the Traveling Salesman Problem Using New Operators in Genetic Algorithms, American Journal of Applied Sciences 6 (8): 1586-1590, 2009