An Improved Genetic Algorithm to Solve the Traveling Salesman Problem

Conference Paper · January 2009

CITATIONS
21

READS
105

2 authors:

Omar M. Sallabi
University of Benghazi
17 PUBLICATIONS **49** CITATIONS
SEE PROFILE

Younis Elhaddad
University of Benghazi
9 PUBLICATIONS **39** CITATIONS
SEE PROFILE

An Improved Genetic Algorithm to Solve the Traveling Salesman Problem

Omar M. Sallabi , Younis El-Haddad

Abstract—The Genetic Algorithm (GA) is one of the most important methods used to solve many combinatorial optimization problems. Therefore, many researchers have tried to improve the GA by using different methods and operations in order to find the optimal solution within reasonable time. This paper proposes an improved GA (IGA), where the new crossover operation, population reformulates operation, multi mutation operation, partial local optimal mutation operation, and rearrangement operation are used to solve the Traveling Salesman Problem. The proposed IGA was then compared with three GAs, which use different crossover operations and mutations. The results of this comparison show that the IGA can achieve better results for the solutions in a faster time.

***Keywords*—** AI, Genetic algorithms, TSP.

I. INTRODUCTION

The Traveling Salesman Problem (TSP) is a classic case of a combinatorial optimization problem. The TSP is one of the most widely known Non deterministic Polynomial (NP hard) problems [1,2]. The TSP is stated as: for a given complete graph, G, with a set of vertices, V, a set of edges, E, and a cost, c_{ij} associated with each edge in E. The value c_{ij} is the cost incurred when traversing from vertex, $i \in V$ to vertex, $j \in V$. Given this information, a solution to the TSP must return the cheapest Hamiltonian cycle of G[7]. Current genetic algorithms may not produce optimal solutions, and if they do, they will not be within a reasonable time[3]. This paper introduces a new crossover technique for genetic algorithms, as well as new operations, which can be effective for fast convergence, and can obtain the best results in a short time period. The improved GA (IGA), which uses these operations, was tested using different datasets of symmetric TSP from TSPLIB [4]. The results indicate that the algorithm provided good results within reasonable time.

O. Sallabi is an assistant professor with the Garyounis University, Faculty of information Technology, Benghazi, Libya, phone (218-91-3828237) , e-mail: osallabi@garyounis.edu.

Y. EL- Haddad , is research assistant with the Garyounis University, Faculty of information Technology, Benghazi, Libya, phone (218-91-3260972) , e-mail: younis_hadded@garyounis.edu.

II. GENETIC ALGORITHMS

The basic principles of genetic algorithms (GAs) were introduced by Holland [5]. The GAs are an optimization and search technique, which emerged from a study of biological evolution. GAs operate on "populations" of potential solutions. which are usually referred to as "chromosomes." Each chromosome represents a set of parameters for a given problem. The chromosomes evaluate to represent the best solutions for a recombination process, which produces new chromosomes. The new, improved chromosomes replace those with poorer solutions. In this way, each new generation becomes closer to the optimal solution. This continues for many generations until the termination condition is met. Mutations and different combining strategies ensure that a large range of search space is discovered [5].

III. IMPROVED GENETIC ALGORITHM (IGA)

Crossover is the most important operation of a GA because in this operation, characteristics are exchanged between the individuals of the population. In accordance, the IGA is concerned with this operation more than population size. Thus, the initial population only consists of two individuals. Applying population reformulates the operation. A Swapped Inverted Crossover (SIC) is applied to these individuals to produce 12 children with different characteristics inherited from their parents. Ten copies are made of these children by applying multi mutation, where each copy mutates with a different method. The fitness function for each individual is evaluated, and the best two selected. Finally, a partial local optimal mutation operation for the next generation is applied. The technique used for the IGA is: the tour is divided to three parts, with two cut points ($\boldsymbol{p_1}$ **and** $\boldsymbol{p_2}$**),** the head containing $(\boldsymbol{1, 2 \ldots\ldots, p_1 - 1})$, the middle containing $(\boldsymbol{p_1, p_1 + 1, \ldots\ldots, p_2})$, and the tail containing $(\boldsymbol{p_2 + 1, p_2 + 2, \ldots\ldots\ldots, n})$. The SIC changing the head of the first parent with the tail of the second parent. The middle remains unchanged, until the partial local optimal mutation operation is applied. This will improve the middle tour by finding its local minima. The role of the population reformulates operation is to change the structure of the tour by exchanging the head and the tail with the middle. This technique ensures that the new cities will be at the middle part of each cycle, ready for improvement.

A. Swapped Inverted Crossover (SIC)

This new crossover is used to improve the GA performances. The main idea of the SIC is to backtrack different ways to search for better tours. This crossover can be applied with a one or two point crossover, or both, as is done here.

Two point SIC - The basic principle of this crossover is two random cut points (p_1 and p_2), a head, containing $(1,2 \ldots\ldots, p_1 - 1)$, the middle containing $(p_1, p_1 + 1, \ldots\ldots, p_2)$, and the tail containing ($p_2 + 1, p_2 + 2, \ldots\ldots\ldots, n$) .The head and tail of each parent are flipped, and then the head of the first parent is swapped with the tail of the other parents, and vice versa. For example, consider parent tours:

Parent1 (1 2 3 4 5 6 7 8 9)
Parent2 (7 4 1 9 2 5 3 6 8)

Suppose that the two selected crossover points are (4 and 6), then the head of Parent1 will be flipped and become **(3 2 1).** In addition, the tail will be **(9 8 7).** The same will be done for Parent2, thus the head will be **(1 4 7)** and the tail **(8 6 3).** The elements of Parent2 that match the head and tail of Parent1 will be removed from Parent2. This is so the remaining elements of Parent2 **(4 5 6)**, the producing offspring, will be formulated with different methods so that each child has the different characteristics of his parents. For example, if the flipped head of Parent1 is swapped with the flipped tail of Parent2, and the flipped tail of Parent1 is swapped with the flipped head of Parent2, the children O1 and O2 will be produced:

O1 →(9 8 7 4 5 6 3 2 1)
O2 → (8 6 3 2 5 9 1 4 7)

If the flipped head of Parent1 is swapped with the flipped head of Parent2, and the flipped tail of Parent1 is swapped with the flipped tail of Parent2, the children O3 and O4 will be produced:

O3→ (3 2 1 4 5 6 9 8 7)
O4→ (1 4 7 2 5 9 8 6 3)

One point SIC- Selecting one crossover point randomly, the head of Parent1 will be flipped and swapped once with the head of Parent2, and second with the tail of Parent2. Suppose that the crossover point is 4, then the head of Parent1, after flipping will be **(4 3 2 1).** Next, these cities are removed from Parent2, so Parent2 will be **(7 9 5 6 8),** and O5 and O6 will be:

O5 → (4 3 2 1 7 9 5 6 8)
O6 → (7 9 5 6 8 4 3 2 1)

The same procedure is done for Parent2, so that the flipped head of Parent2 is **(9 1 4 7)** and that of Parent1 is **(2 3 5 6 8).** The produced offspring, O7 and O8 will be:

O7→ (9 1 4 7 2 3 5 6 8)
O8→ (2 3 5 6 8 9 1 4 7)

Now, applying the same procedure, but using the tail instead of the head, and having the same crossover point of 4, the tail of Parent1 after flipping is **(9 8 7 6 5 4).** After removing these tours, Parent2 will **be (1 2 3).** Inserting the inversed tail of Parent1 as the head of Parent2 produces O9, and inserting the inversed tail of Parent1 as the tail of Parent2 will produce O10:

O9 → (9 8 7 6 5 4 1 2 3)
O10 → (1 2 3 9 8 7 6 5 4)

For Parent2, the flipped tail is **(8 6 3 5 2 9)** and for Parent1 it is **(1 4 7),** so the produced children are:

O11→ (8 6 3 5 2 9 1 4 7)
O12 → (1 4 7 8 6 3 5 2 9)

B. The Rearrangement Operation

A new operation called Rearrangement is applied to all populations (see figure1). $c_{i,j}$, is the cost between the two adjacent cities $city_i$ and $city_j$, where $i = 1,2,3, \ldots\ldots, n-1$ and $j = i + 1$, . The aim of this operation is to find the greatest (max) value of $c_{i,j}$ among all the adjacent cities on the tour, and then swap $city_i$ with three other cities, one at a time. These cities are located on three different positions on the tour (beginning, middle, and the end). The best position, plus the original position will be accepted. Since this operation works in random matter, it may not achieve any improvement after several iterations, but it may take big jump and improve the result.

$i = 1$
$S0 = S$
$max = 0$
While $i < n - 1$
  If $c_{i,i+1} > max$
    Begin
      $Max = c_{i,i+1}$
      $X = i$
    End
$S1 \leftarrow$ Swap $city_x$ with $city_1$
$S2 \leftarrow$ Swap $city_x$ with $city_{\frac{n}{2}}$
$S3 \leftarrow$ Swap $city_x$ with $city_n$
$S \leftarrow min\,(S0, S1, S2, S3)$
Return (S)
End

Figure 1: The rearrangement operation procedure

C. Multi mutation operation

In this operation, the selected parents are copied 10 times, and each individual of these copies is mutated using a different nearest neighbor. For a better chance of success, two random cities are selected. Suppose these cities are c_i and c_j, then ten different mutation criteria can be applied, one for each copy: 1) swap c_i with c_{i+1}; 2) swap c_i with c_{i-1}; 3) swap c_i with c_{i+2}; 4) swap c_i with c_{i-2}; 5) swap c_j with c_{j+1}; 6) swap c_j with c_{j-1}; 7) swap c_j with c_{j+2}; 8) swap c_j with c_{j-2}; 9) swap c_i with c_j; and 10) swap c_1 with c_n, where $2 < i, j < n - 2$, and n is number of cities.

D. Partial local optimal mutation operation

In this operation, the subtour of individuals within the range of $(3 \leq size\ of\ subtor < n/2$) is selected randomly. Then we find the tour that produces the local minima of this subtour and exchange it with the original subtour. This operation is undertaken on one of the selected individuals after the multi mutation operation is performed.

E. Population reformulates operation

It is known that a tour can be represented as a Hamiltonian cycle of a graph[7]. Hence, we can reformulate it without changing its fitness value. For example, the tour

1 2 3 4 5 6 7 8 9

can be reformulated as: cut the tour into two equal parts and reverse each one: 5 4 3 2 1 and 9 8 7 6. Then, reconnecting the resulted subtours, the reformulated tour will be

5 4 3 2 1 9 8 7 6

This operation is used to refresh the population in order to prevent the algorithm from being trapped at a local minimum.

Table 1 shows the experiments undertaken to select the optimum size of population for the IGA. Using the known instance, kroA100, for which its optimal solution is 21282 for each population size, 10 runs are done, and 60 seconds is the maximum time for each run. The error is calculated according to the equation ,

$$Error\ = \frac{average\ -\ optimal}{optimal} \times 100.$$

and column 5 in Table 1 indicates the number of times the optimal solution is obtained from 10 runs.

TABLE 1:
EFFECT OF POPULATION SIZE ON THE PERFORMANCE OF THE IGAS

Population size	Best result	Average	St. deviation	Performance	Error
2	**21282**	**21294.5**	**26.36**	**8/10**	**0.058**
6	**21282**	**21505.9**	**189.8**	**2/10**	**1.05**
10	**21282**	**21589.1**	**325.98**	**1/10**	**1.44**
14	**21282**	**21660.5**	**278.65**	**1/10**	**1.77**
20	**21379**	**21784.8**	**381.98**	**0/10**	**2.36**

IV. EXPERIMENTAL RESULTS

For a fair comparison with the other researchers' works (Ray et al) [8], that used new operations to improve the GA to solve the TSP, it is reasonable to compare the proposed IGA with available research in the same area. By using the same dataset of TSP instances, the number of iterations in the test is the same. Ray et al, propose a new crossover operation called the modified order crossover (MOC), which is based on the order crossover (OCX). They also used a new operation called the Nearest Fragment (NF) Heuristic. The authors referred to their GA as (FRAG_GA), and they compared their results with other GAs that use different crossover methods, such as SWAP_GATSP [6] and OX_SIM [2] We chose instances that are (≥ 100 cities) and used by Ray et al. The IGA is run for 10 trials in each instance. Using Table 2, which was created by Ray et a[8], we added two new columns to shows the results obtained by our IGA.

The best result obtained is shown in column 6 of Table 2. Also, the number of iterations required to find it is written in parentheses. Column 10 shows the average result obtained from 10 trials, as well as the number of iteration in parentheses.

TABLE2.
RESULTS USING FRAG_GA, SWAP_GATSP, OX_SIM AND IGA FOR DIFFERENT TSP INSTANCES

		Best Result				Average Result			
Problem	Optimal	FRAG_GA	SWAP_GATSP	OX_SIM	**IGA**	FRAG_GA	SWAP_GATSP	OX_SIM	**IGA**
KroA100	2182	21282 (800)	21504 (5000)	22400 (25000)	**21282 (454)**	21350 (2000)	21900 (5000)	22670 (30000)	21294.5 (2000)
D198	15780	15834 (3000)	15992 (7000)	16,720 (25000)	15781 (2514)	15964 (3500)	16,132 (10000)	18200 (40000)	15799.3 (3500)
Pcb442	50778	51104 (8000)	52620 (15000)	53402 (40000)	51101 (9998)	51930 (10000)	53,820 (20000)	59740 (65000)	51208 (10000)
Rat783	8806	9007 (15000)	9732 (30000)	10810 (70000)	8924 (15000)	9442 (20000)	10110 (40000)	11520 (10000)	9216 (20000)

V. CONCLUSION

This paper proposed IGA which can be effective to solve TSP with its new crossover SIC and the new operations described above. All these operations together able to traverse the search space and produce good results. It is clear from Table 2 that for different TSP instances, the IGA has a better performance than the three compared GAs.

REFERENCES

[1] **E. Lawle, ,** "*Combinatorial Optimization: Networks and Matroids".*, Holt, Rinehart, and Winston, New York1. 1976.

[2] **P. Larranaga,C.M.H. Kuijpers, R.H. Murga, I. Innza and S. Dizdarevic** *,"Genetic Algorithms for the Travelling Salesman Problem: A Review of Representations and Operators".* **P..** 13(2), s.l. : Artif. Intell, 1999. 129-170.

[3] **D. Beasley, D. Bull, and R. Martin.** "*An Overview of Genetic Algorithms: Part 1. Fundamentals",* University Computing, 1993, Vol. 15(2), pp. 58--69.

[4] TSBLIB, http://www.iwr.uni-eidelberg.de/iwr/ comopt/ soft/ TSPLIB95/TSPLIB.html. [Online] [Cited: 12 22, 2008.]

[5] **Holland, J.**,"*Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence."* The University of Michigan Press, 1975.

[6] **S. Ray, S. Bandyopadhyay and S. Pal, "***New operators of genetic algorithms for traveling salesman problem"* Cambridge : s.n., 2004, Vol. 2.

[7] **M. Bakhouya and J, Gaber "** *An Immune Inspired-based Optimization Algorithm: Application to the Traveling Salesman Problem",*. : Advanced Modeling and Optimization , 2007 , Vol. 9.

[8] **S. Ray, S. Bandyopadhyay and S. Pal, "** *New Genetic operators for solving TSP: Application to microarray gene ordering",* Berlin : Springer, 2005.

