Three Genetic Solutions for Travel Salesman problem

Hanan A.
Technical Institute in Najaf/ Computer Department

ABSTRACT

ABSTRACT

The classic Traveling Salesman Problem **(TSP)** has been frequently used as a test bed for the study of new local search techniques developed for general circuit-based permutation problems, local search heuristics for the traveling salesman problem (TSP) is chiefly based on algorithms using the classical Lin-Kernighan (L-K) procedure, La Lena simple genetic algorithm. This work give a detailed description of the three genetic solutions for this problem and found that genetic algorithms different in behavior according to the tour length.

[illegible]

INTRODUCTION

The traveling salesman problem (TSP) is a well-known NP-hard combinatorial optimization problem [1]. It can be stated as follows. There are C cities, which are numbered from 0 to $C - 1$. The distance from city i to city j is known to be dij, where $0 \leq i, j < C$ and $dij = 0$ if $i = j$. A tour is a path that starts from a city, visits each city exactly once, and goes back to the starting city. Mathematically, a tour can be expressed by a vector t of C elements. Each of the elements in t represents a city, i.e., $0 \leq t(i) < C$ and $t(i) \neq t(j)$ if $i \neq j$.

The tour starts from $t(0)$, visits cities in the order they appear in t, and then goes back to $t(0)$ after visiting $t(C - 1)$. The goal of the TSP is to find a tour t with the minimum tour length, i.e., to minimize

$$\sum_{i=0}^{C-2} d_{t(i)t(i+1)} + d_{t(C-1)t(0)}$$

Standard genetic algorithm

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. Table (1) show simple genetic algorithm

Table(1) simple genetic algorithm

Parent 1	F A B | E C G D
Parent 2	D E A | C G B F
Child 1	F A B | C G B F
Child 1	D E A | E C G D

The difficulty with the Traveling Salesman Problem is that every city can only be used once in a tour. If the letters in the above example represented cities, this child tours created by this crossover operation would be invalid. 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 [2] 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.

First algorithm
Stores the **links** in both directions, for each tour. In the above tour example, Parent 1 would be stored as shown in table (2) bellow:

Table (2) links for each node

City	1ST Connection	2ND Connection	3RD Connection	4TH Connection	5 TH Connection	6TH Connection
A	B	C	D	E	F	G
B	A	C	D	E	F	G
C	A	B	D	F	E	G
D	A	B	C	E	F	G
E	A	B	C	D	F	G
F	A	B	C	D	E	G
G	A	B	C	D	E	F

The crossover operation is more complicated than combining 2 strings. Using modified two swap (M2S) given in [3]with link information described in table can retain link information from parents to the offspring for example if we have two tours **AFDGCEB**, and **GCEBAFD**, then choose two genes randomly from parent1 and swap them by corresponding genes in parent 2 if two genes **F** and **C** will be selected randomly then the new children be:
ACDGFEB
GFEBACD

SECOND ALGORITHM

David Goldberg proposed the "partially-mapped crossover" operator [4] : given two parent tours say:
A= 9 8 4 5 6 7 1 3 2 10
B= 8 7 1 2 3 10 9 5 4 6

the child are produced by picking two random numbers and swapping the cities within the bounds. Some clean up is then required because might be the child has duplicate cities. if a city is represented twice in the child simply replace the first occurrence with the city that got swapped away:

for example if the two random numbers are 4 and 6, the substrings to swap are "567" from a and "231" from b resulting in:

child1=671|567|9546
now swap the other 5 with 2, the 6 wit a 3 and 7 with 10

child1=8 10 1 | 5 6 7 | 9 2 4 3
which a valid tour

Third algorithm

Whitley's representation [5] builds on the adjency of they believe that since the edges are the important component of TSP, that they should be encoded on the genome instead of the ordering of the cities. In addition, the developed a crossover operator called edge mapped crossover which transfares 95% of the edges from parents to Childs

for example for two tours **(a b c d e f)** and **(bdcaef)**
the edges be as:
a:bfce
b:acdf
c:bda
d:ceb
e:dfa
f:aeb

to create a child we can start from any node and build a tour based on it's edges beginning from the city with minimum of edges so:
BCDEAF is one of Childs.
The power of this algorithm is the incorporation of the parent information to the child by sharing subtours of the parents.

Implementation

These three procedures had been implemented with three scenarios small, medium, and large tours of about 5, 20, 50 cities. Figures 1, 2, 3 express the results of the three algorithms respectively.

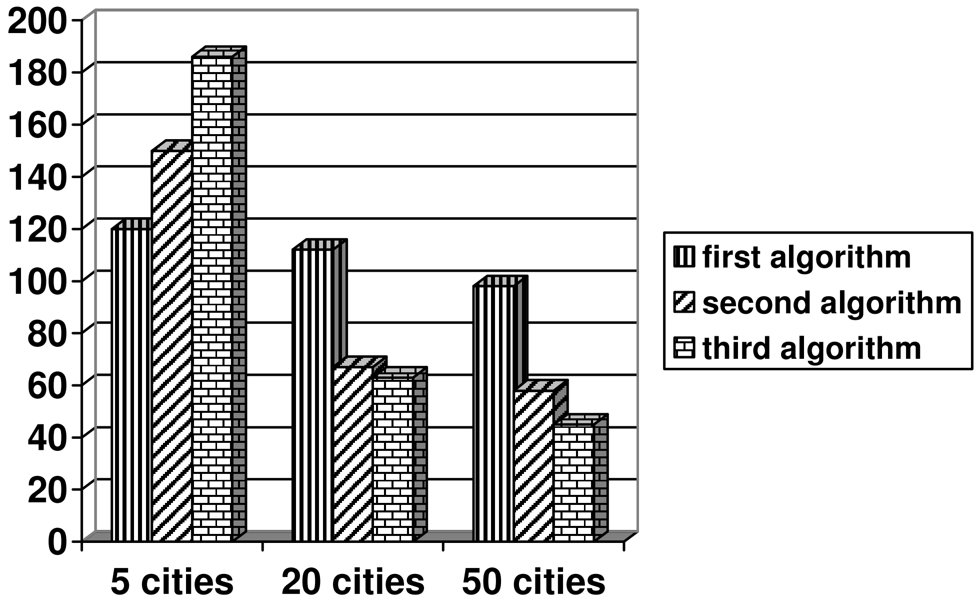

Figure 1

Conclusion

From figure 1 we see that first algorithm work well with small scenario while its performance degraded this result from that it 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. 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. While the second algorithm and the third one made better, but in small scenario it is obvious that these algorithms spend long time in cleanup the chromosomes and in reordering of the cities respectively.

References

[1] E. L. Lawler, J. K. Lenstra, and D. B. Shmoys, *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*. New York, NY: Wiley-Interscience, 1985.

[2] John Grefenstette, Rajeev Gopal, Brain Rosmaita, Dirk Van Gucht, "genetic algorithms for the traveling salesman problem" international conference on genetic algorithms, 1985.

[3] S. Lin and B. Kernighan, An Effective Heuristic Algorithm for the Traveling Salesman Problem, Operations Research 21 (1973) 498-516.

[4]David Goldberg and Robert Lingle, " alleles, loci, and traveling salesman problem", international conference on geneticalgorithms,1985

[5] Darrell Whitley, Tim Starkweather and D' Ann Fuquay "scheduling problems and travel salesman: the genetic edge recombination operator" international conference on genetic algorithms, 1989.