Genetic Algorithm Solution of TSP Using PMX Crossover

Milena Karova[1] Blagovest Razhenkov[2] Julka Petkova[3]

Abstract – **This paper presents implementation of Genetic Algorithms (Gas) and genetic operators to find optimal solution of Travel Sales Problem. The essence of the problem consists of finding a path with the least length. The presented Genetic Algorithm uses random presentation of chromosomes in the population and new type of crossover: PMX. It is suitable for application specific tasks for a 100% certainty that after crossover, newly chromosome doesn't contain repetitive genes. The approach is tested on a set of randomly generated problems.**

Keywords – **Travel Salesman Problem, optimization, crossover, mutation, genetic operators, evolution strategy, PMX crossover.**

I. Introduction

The Traveling Salesman Problem (TSP) is one of the top ten problems, which has been addressed extensively by mathematicians and computer scientists. The classical formulation is stated as: Given a finite set of cities and the cost of traveling from city *i* to city *j*, if a traveling salesman were to visit each city exactly once and then return to the home city, which tour would incur the minimum cost? Formally, the TSP for n nodes may be defined as follows:

$$\min f(\underline{x}) = \sum_{i=1}^{n-1} d_{x_i x_{i+1}} + d_{x_n x_1} \qquad (1)$$

where x is a vector containing a permutation of the set of $\{1,2,3\ldots n\}$, x_i is the ith element of x and d_{ij} is the distance between nodes i and j.

The TSP has become a standard benchmark for combinatorial optimization methods that attempt to find near optimal solutions for NP-hard problems. Several methods, usually based on search heuristics, have been applied including local search [6], simulated annealing [8], tabu search [1] and neural networks. The pure Genetic Algorithm (GA) [4] has a poor performance record when applied to the TSP. This has motivated researchers to construct GA based approaches that incorporate TSP specific heuristics including, different genetic schemes [3], heuristics for generating the initial population [5] and specific operators for crossover and mutation. PMX crossover operator is a variety of double point crossover [2].

[1]Milena N. Karova is with the Department of Computer Science and Technologies, Technical University Varna, Studentska str. 1, 9010 Varna, Bulgaria, E-mail: mkarova@ieee.bg

Blagovest Razhenkov is is a student within the Department of Computer Science and Technologies, Technical University Varna, Studentska str. 1, 9010 Varna, Bulgaria, Email: razhenkov@abv.bg

Julka P. Petkova is the Department of Computer Science and Technologies, Technical University Varna, Studentska str. 1, 9010 Varna, Bulgaria, E-mail: jppet@abv.bg

II. PMX Crossover

A. Definition

The PMX crossover is known as Partially Mapped Crossover. There are different variants of PMX crossover. The simple PMX chooses a subsequence of a tour from one parent and preserve the order and position of as many cities as possible from the other parent. If the parents are:

p1=123*5467*89
p2=452*1876*93

the offspring will be generated as follows:

c1=(xxx*1876*xx).

This swap also defines a mapping 1-4, 8-5, 7-6, 6-7. The next step is to fill cities from the parents that don't lead to conflict.

c1=(x23*1876*x9)

The remaining positions are filled with cities that do not appear in the new chromosome

c1=(423*1876*59)

The presented type of PMX crossover is suitable for application specific tasks for a 100% certainty that after crossover, newly chromosome doesn't contain repetitive genes. The entire procedure is explained in Fig. 1. There are the rules to choose each gene, its index and corresponding gene from the other parent. This approach increases the diversity of Genetic Algorithm.

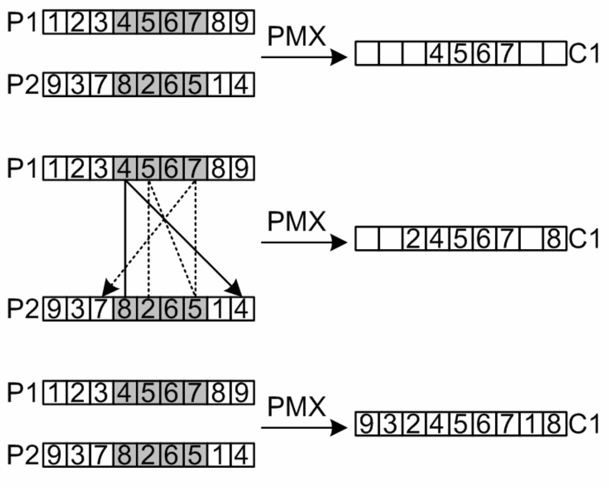


Fig. 1. PMX Crossover

B. PMX crossover and TSP Problem

From the first parent it takes random part of its gene and directly copied into a new chromosome. Then the positions of which are taken from the parent gene is monitored in the second, and if these items have a gene, which has not yet been copied into the new chromosome, it is the following steps:

- Take the position of a gene located in the second parent. Looking for the corresponding gene at the same position of first parent
- Looking for the same gene in the second parent.
- If the position of the gene in the second parent is part of the new chromosome, back to point 1.
- If the position of the gene in the second parent is not yet part of the new chromosome, put it in this position in the new chromosome.

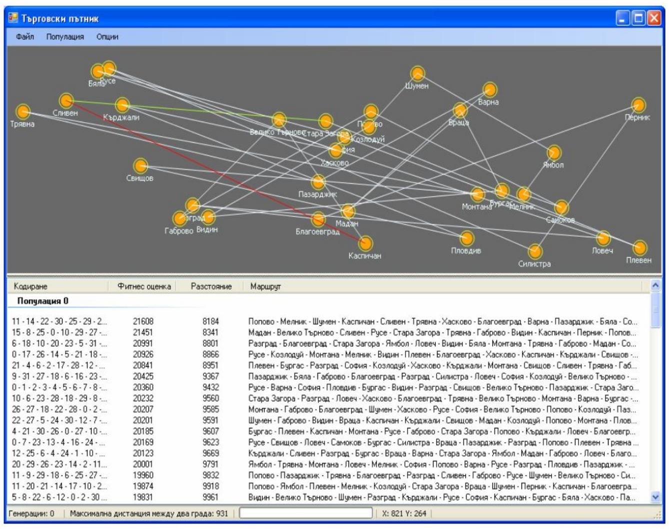
III. Software Design

A. Functional description

The GA system must evolve a solution to the TSP problem in the form of finding the minimum cost [7] (1).

B. System Overview

Software application allows the user to manually or automatically generated cities. In the case of automatically generation, the cities number is specified in the settings of the program. As manual method with a single click the user creates city in the gray panel. Every city has a name (names are up to 64 characters). The beginning of journey is colored in green and the last path – in red. (Fig. 2).



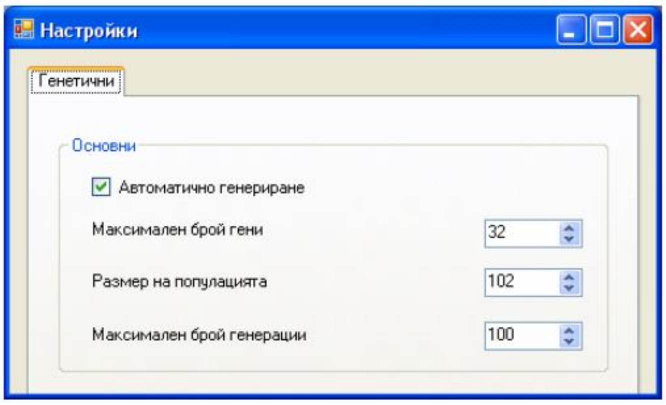
Кодиране	Фитнес оценка	Разстояние	Маршрут
Популация 0			
11 - 14 - 22 - 30 - 25 - 29 - 2...	21608	8184	Попово - Мелник - Шумен - Каспичан - Сливен - Трявна - Хасково - Благоевград - Варна - Пазарджик - Бяла - Со...
15 - 8 - 25 - 0 - 10 - 29 - 27 -...	21451	8341	Мадан - Велико Търново - Сливен - Русе - Стара Загора - Трявна - Габрово - Видин - Каспичан - Перник - Попов...
6 - 18 - 10 - 20 - 23 - 5 - 31 -...	20991	8801	Разград - Благоевград - Стара Загора - Ямбол - Ловеч - Видин - Бяла - Монтана - Трявна - Габрово - Мадан - Со...
0 - 17 - 26 - 14 - 5 - 21 - 18 -...	20926	8866	Русе - Козлодуй - Монтана - Мелник - Видин - Плевен - Благоевград - Хасково - Каспичан - Кърджали - Свищов -...
21 - 4 - 6 - 2 - 17 - 28 - 12 - ...	20841	8951	Плевен - Бургас - Разград - София - Козлодуй - Хасково - Кърджали - Монтана - Свищов - Сливен - Трявна - Габ...
9 - 31 - 27 - 18 - 6 - 16 - 23 -...	20425	9367	Пазарджик - Бяла - Габрово - Благоевград - Разград - Силистра - Ловеч - София - Козлодуй - Велико Търново - ...
0 - 1 - 2 - 3 - 4 - 5 - 6 - 7 - 8 -...	20360	9432	Русе - Варна - София - Пловдив - Бургас - Видин - Разград - Свищов - Велико Търново - Пазарджик - Стара Заго...
10 - 6 - 23 - 28 - 18 - 29 - 8 -...	20232	9560	Стара Загора - Разград - Ловеч - Хасково - Благоевград - Трявна - Велико Търново - Монтана - Варна - Бургас -...
26 - 27 - 18 - 22 - 28 - 0 - 2 -...	20207	9585	Монтана - Габрово - Благоевград - Шумен - Хасково - Русе - София - Велико Търново - Попово - Козлодуй - Паз...
22 - 27 - 5 - 24 - 30 - 12 - 7 -...	20201	9591	Шумен - Габрово - Видин - Враца - Каспичан - Кърджали - Свищов - Мадан - Козлодуй - Попово - Монтана - Плов...
4 - 21 - 30 - 26 - 0 - 27 - 10 -...	20185	9607	Бургас - Плевен - Каспичан - Монтана - Русе - Габрово - Стара Загора - Попово - Кърджали - Ловеч - Благоевгр...
0 - 7 - 23 - 13 - 4 - 16 - 24 - ...	20169	9623	Русе - Свищов - Ловеч - Самоков - Бургас - Силистра - Враца - Пазарджик - Разград - Попово - Плевен - Трявна ...
12 - 25 - 6 - 4 - 24 - 1 - 10 - ...	20123	9669	Кърджали - Сливен - Разград - Бургас - Враца - Варна - Стара Загора - Ямбол - Мадан - Габрово - Ловеч - Благо...
20 - 29 - 26 - 23 - 14 - 2 - 11...	20001	9791	Ямбол - Трявна - Монтана - Ловеч - Мелник - София - Попово - Варна - Русе - Разград - Пловдив - Пазарджик - ...
11 - 9 - 29 - 18 - 6 - 25 - 27 -...	19960	9832	Попово - Пазарджик - Трявна - Благоевград - Разград - Сливен - Габрово - Русе - Шумен - Велико Търново - Си...
11 - 20 - 21 - 14 - 17 - 10 - 2...	19874	9918	Попово - Ямбол - Плевен - Мелник - Козлодуй - Стара Загора - Враца - Шумен - Перник - Каспичан - Благоевгра...
5 - 8 - 22 - 6 - 12 - 0 - 2 - 30 ...	19831	9961	Видин - Велико Търново - Шумен - Разград - Кърджали - Русе - София - Каспичан - Бургас - Бяла - Хасково - Па...

Генерации: 0 | Максимална дистанция между два града: 931 | X: 821 Y: 264

Fig. 2. TSP solution program interface

Users are given the opportunity to monitor overall process of finding the most optimal solution. In the list information the population is indicated as "population" with its serial number.

In the menu Encoding the population of chromosomes is presented and their genes. In the menu Fitness evaluation the chromosome fitness is displayed. Distance column gives possibility to observe the variation of travel distance (i.e chromosome). The last column Route shows the names of cities ranked in order as be visited. During the whole process it is displayed all possible routes.



Автоматично генериране ✓

Максимален брой гени 32

Размер на популацията 102

Максимален брой генерации 100

Fig. 3. GA's settings

When the predefined number of generations is over, the message box appears with the best available route. Software application makes it possible to graphically see the best and worst case route. The user is able to export the chart of the best route in .jpg format, and the route in a text file.

IV. Genetic Algorithm

A. Settings. Initial parameters

In the section Settings (Fig. 3.) the user can set a maximum number of genes, population size and maximum population size. Initial parameters are:
Crossover rate: 0,7-0,9
Crossover type: PMX, Single-point
Mutation rate: 0-2,5%
Population size:varies
Chromosome Length: 32 up 120
Measurements: Sum, Average

B. Encoding, initial population, fitness function, selection

Permutations of the range [0, N-1] are encoded as a series of N integers in the chromosome. Each integer represents the ID number of city.

The GA system uses random generation of initial population with repair function (if there are the similar cities). So the diversity of the population is greater and the probability to find an optimal solution increases.

The GA system attempts to minimize the distance value shown at Eq. 1

The using selection method is rank-based selection. Initially the fitness function f_i is calculated for every individual; the average of the function of the target population f_{sr} as the average target values for all subjects. Then, for each individual is calculated ratio f_i / f_{sr}. If this is greater than 1, the individual has a great life and be allowed to cross, otherwise the individual is removed. By using this type of selection all the chromosomes have a real chance to be selected.

C. Crossover and mutation

The system used a variant of PMX, presented in II A. Random index are generated between *0* and *N-1*. The permutation elements at that index are recorded from both parents. Those elements are swapped among both chromosomes (Fig. 1). This process was repeated *N* times.

PMX crossover delays the performance of GA, but it ensures the minimum number of identical chromosomes in the population.

Here inversion mutation (IM) is performed on each string as follows:

The mutation operator selects randomly two cut points in the string, and it reverses the substring between these two cut points. For example consider the tour.

(1 2 3 4 5 6 7 8),

and suppose that the first cut point is chosen randomly between 2nd city and 3rd city, and the second cut point between the 5th city and the 6th city. Then the resulting strings will be

P = (1 2 |3 4 5| 6 7 8)

C = (1 2 |5 4 3| 6 7 8)

The mutation probability it is not kept constant over the generations. Rather it is varied in cycles of appropriate intervals (linearly from 0 to 2,5%).

The time complexity of algorithm is given by *O(k*N*n)*, where *k* is the numbers of generation, *N* is the size of population and *n* is the data size or the number of cities.

V. Implementation and Results

Experiments were designed to compare the classical crossover and PMX crossover, to change the number of generation (using PMX crossover) and to explore the process of mutation operator.

In the first case the size of the population changes in the same number of generations, same number of cities and the probability of crossover ranged between 0.7 and 0.9.

Fig. 4 shows that using PMX crossover, it receives good results than using single-crossover. City's number is 32.

The optimal solution appears at N=32. When the population size increases, the distance (fitness function) became worse.

If it changes number of generations, 50-60 is optimal number (the minimum fitness value is achieved) (Fig. 5).

The TSP problems from this solution for which optimal results were obtained are shown in Table 1. The algorithm, in its current form, is able to optimally optimize TSP problems where the number of cities is less than 100.

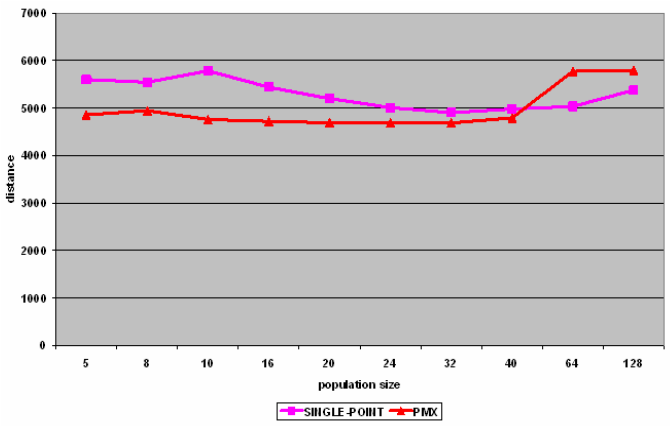


Fig. 4. PMX and single-point crossover

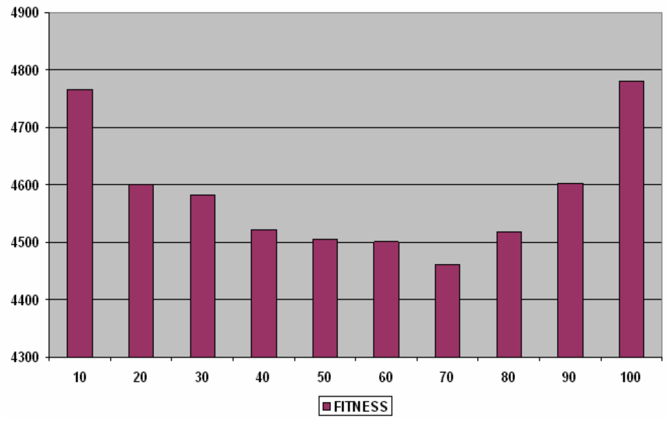


Fig. 5. Change the number of generations

TABLE I

N of cities	Optimal Tour	Generation
10	2450	17
20	2980	24
30	4590	20
40	5300	31
50	5933	49
60	6990	34
70	7348	54
80	8042	73
90	8692	57
100	9100	160
110	9800	150

The performance of mutation operator is very interesting. The mutation operator is effective when the distances are large. However, as the genetic algorithm progresses, the mutation operator became less beneficial (Table II).

TABLE II

Optimal distance	0% mutation	2,5% mutation
2349	2406	3533
3777	3980	4102
4560	4690	4535
7735	7798	7742
9771	9700	9546
10888	11904	10820

VI. CONCLUSION

The results obtained with these genetic operators are impressive, on practical data set. The method using PMX crossover can be easily adapted to solve the other optimization tasks.

There is difference between a local optimization heuristics and genetic algorithm, developed for TSP. Local optimization techniques were used to find local minimum in the search space and the genetic algorithm used to search the space of local minimum to find the basin in which the global minimum is located.

There are several issues for future research: firstly - to mix divers heuristic techniques with genetic algorithms; secondary – to create parallel variant of these solution.

Genetic algorithm enables the use of different genetic operators in the search for optimal solution. The priorities of PMX crossover are two: 1) do not allow the repetition of cities in a chromosome; 2) it obtains better results compared with single-point crossover; 3) the number of identical chromosomes in the population is minimized.

REFERENCES

[1] Fiecheter L., A Parallel tabu Search Algorithm for Large Travelling Salesman Problems, Discrete Applied Mathematics and Combinatorial Operations Research and Computer Science, 1994

[2] Freisleben B., Merz P., A Genetic Local Search Algorithm for Solving Symmetric and Asymetric Travelling Salesman Problems, IEEE International Conference on Evolutionary Computation, 1997

[3] Karova M., Petkova J., Penev S., Web Application of Traveling Salesman Problem using Genetic Algorithms, Proceedings of Papers, volume2, XLII International Scientific Conference on Information, Communication and energy Systems and Technologies ICEST'2007, June 24-27 2008, Ohrid, Macedonia, pp. 849-852.

[4] Karova M., Genetic Algorithms And Genetically Operator Crossover, Proceedings of the International Conference on Computer Systems and Technologies, CompSysTech'2001, Sofia, Bulgaria, 21-22 june 2001, pp.II24-1-II.24-5.

[5] Lalena M., TSP solver, http://www.lalena.com/ai/tsp

[6] Reinelt G., The Travelling Salesman: Computational Solutions for TSP Application", Lecture Notesin Computer Science, 840, Springer-Verlag

[7] TSPLIB HomePage: http://www.iwr.uniheidelberg.de/groups/comopt/software/TSP

[8] van Laarboven P., Simulated Annealing: Theory and Applications, Kluwwer Academy Publishers, 1987.