ISSN NO. 2320-5407

Journal Homepage: -www.journalijar.com

INTERNATIONAL JOURNAL OF ADVANCED RESEARCH (IJAR)

Article DOI:10.21474/IJAR01/6406
DOI URL: http://dx.doi.org/10.21474/IJAR01/6406

RESEARCH ARTICLE

EVALUATION OF THE OPERATORS OF THE GENETIC ALGORITHM IN APPLICATION ON THE TRAVELING SALESMAN PROBLEM.

S. Bourazza.
Mathematics Department, faculty of Science, Jazan University, KSA.

Manuscript Info

Manuscript History

Received: 01 December 2017
Final Accepted: 03 January 2018
Published: February 2018

Keywords:-
Genetic Algorithm, Traveling Salesman Problem.

Abstract

In this paper, we applied different operators of crossover and mutation of the genetic algorithm to solve the Traveling Salesman Problem (T.S.P.). We will compare them and give the best operators for the crossover and for the mutation with taking in account the effect of the interaction.

Introduction:-

Formulation of the problem:-
The traveling salesman problem (T.S.P.), consists in finding a minimum distance tour of V points (cities, renting, warehouses,...), starting and ending at the same city and visiting each other city exactly once.

In graph theory, the problem can be defined on a graph (X, A), where $X = \{v_1, v_2, \dots, v_V\}$ is a set of V vertices (nodes) and $A = \{(v_i, v_j) \mid v_i \in X, v_j \in X \text{ and } i \neq j\}$ is a set of arcs, together with a non-negative cost (or distance) matrix associated with A. The problem is considered to be symmetric (STSP) if for all. Elements of A are often called edges. The version of STSP in which distances satisfy the triangle inequality is the most studied special case of the problem. The STSP consists in determining the Hamiltonian cycle, often simply called a tour, of minimum cost.

In other words, a T.S.P. with size V is defined by a set of points $v = \{v_1, v_2, \dots, v_V\}$ for which we define a function d.

A solution in the T.S.P. corresponds to $v_{\prod} = (v_{\prod(1)}, v_{\prod(2)}, \dots, v_{\prod(V)})$ where $\prod$ is a permutation on $\{1,2,\dots, V\}$.
Every V-tuple represents a solution for the T.S.P problem.
The objective function associated with T.S.P. which determines how good a solution is. It calculates the total cost of edges in a solution.

$$f(v_{\prod}) = \left(\sum_{i=1}^{i=V-1} d(v_{\prod(i)}, v_{\prod(i+1)})\right) + d(v_{\prod(V)}, v_{\prod(1)})$$

Corresponding Author:-S. Bourazza.
Address:-Mathematics Department, faculty of Science, Jazan University, KSA.

Genetic Algorithm:-
The genetic algorithms are part of the family of the *evolution algorithms*. They are inspired by the creed of the nature *the survival is the best adapted individual to the environment*.

These algorithms have for particularity to be traced on the evolution natural. To these notions of evolution, we add some properties observed in genetics (crossover, mutation, selection, ...)

from where the name of genetic algorithms. They caused the interest many researchers starting with Holland ([Hol-75]), that developed the fundamental principles, passing by Goldberg ([Gol-89]) that used them to solve some concrete problems of optimization. The other researchers followed this way notably Davis ([Dav-93]),
Mahfoud (([Mah-92]) and ([Mah-95])), Michalewicz (([Mich-96]) and ([Mich-2000])), etc.

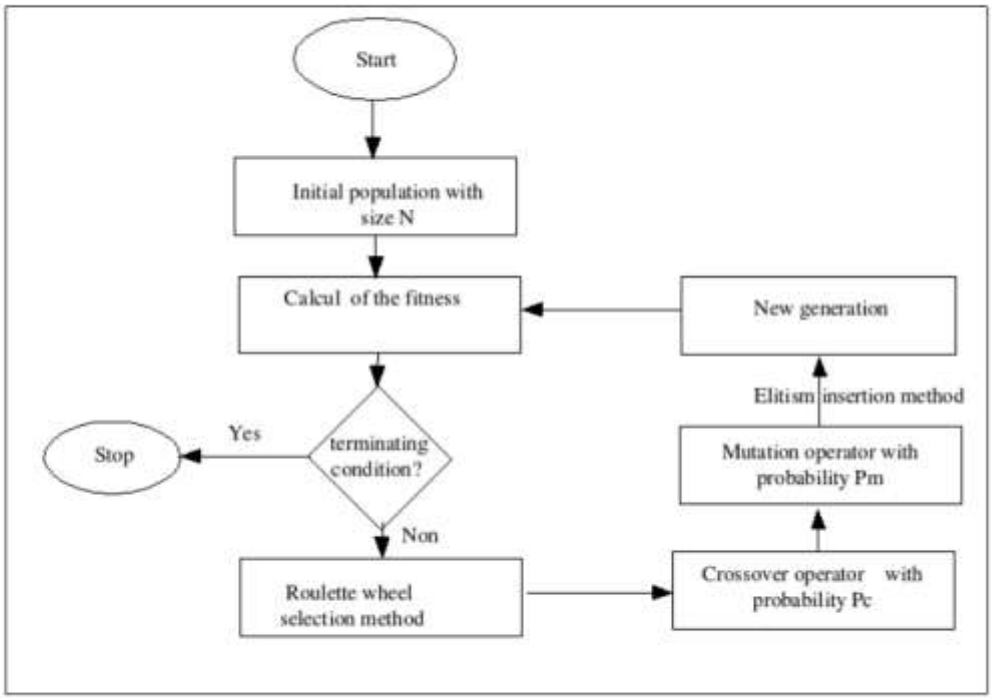

Start
Initial population with size N
Calcul of the fitness
New generation
Elitism insertion method
Yes
Stop
terminating condition?
Mutation operator with probability Pm
Non
Roulette wheel selection method
Crossover operator with probability Pc

Figure 1:-The organization of our genetic algorithm.

Our algorithm, presented in the figure 1, start by generating an initial population of N individuals, for which, we calculate their values of the function objective and we select the better adapted individuals associated with the principle of the roulette wheel selection. The individuals, topics of recombination by the crossover operator, are chosen according to a probability P_c.

Their results can be transferred by an operator of mutation with P_m probability of mutation.

The individuals descended of these genetic operators will be inserted in the new population of which we value the values of the function objective of every individual. After, we met the terminating condition. It consists in seeing if the deviation of the fitness is equal to zero. If the end condition is verified the algorithm then stops with an optimal solution, otherwise one repeats the process on the new generation.

Representation:-
A circuit is coded by a table of integers that represents the successor and the predecessor of every city.

6	2	4	1	5	3	7

Table1:-Coding of the chromosome.

The table 1 represents the following circuit: The starting point is the city 6 that is also the city of arrival, passing by the cities according to their order of apparition in the table:

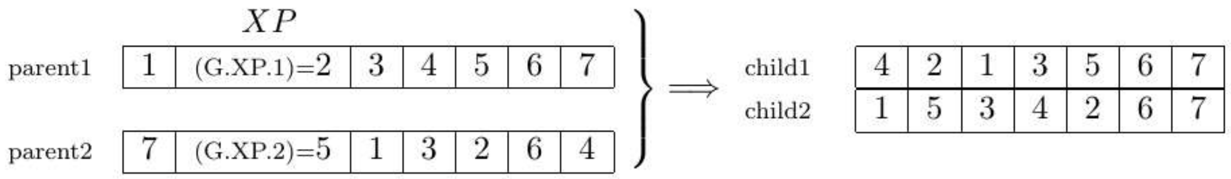
6 → 2 → 4 → 1 → 5 →3 →7 →6

Selection method:-
The used method is roulette wheel selection. It consists in associating to an i chromosome of the population a proportional portion in $1 - \frac{f_i}{\sum_{j \in Population} f_j}$, where f_i is the value of the objective function for the i individual. So, the individuals who have the small fitness can have a strong luck to be accepted and are subjected to the operation of crossover.

Crossover Operators:-
They manipulate the structure of the chromosomes with certain randomness of two solutions called parents in order to produce two solutions called children. It permits to discover the space of the solutions. From the literature, we choose eight operators of crossover that we are going to explain how they proceed for constructing two chromosomes (solutions) named ***child1*** and ***child2*** from two chromosomes ***parent1*** and ***parent2***.

Cyclic operator crossover (cycle by Olivier and al. ([Oli-87])):-
We note by (G.XP.1) the gene at the XP position in the parent1 and by (G.XP.2) the gene of the parent2 located in the XP position.

A position (XP) is chosen at random in parent1, we find the gene (G.XP.1) that we will place in the position (XP) of the child1. After the genes of the parent2, from the position (XP+1), are placed in the child1 until finding the gene (G.XP.1) in the parent2 in a position (XP+k) then (G.XP+k.1) is put in the child1 and we continue to fill this last from parent1 with jumping the genes that have already been introduced. To get the child2, we proceed in the same way. We begin in the same position XP with inverting the roles of the parent1 and the parent2.



		XP					
parent1	1	(G.XP.1)=2	3	4	5	6	7
parent2	7	(G.XP.2)=5	1	3	2	6	4

child1	4	2	1	3	5	6	7
child2	1	5	3	4	2	6	7

Table2:-The cyclic crossing operator (cycle).

Edge recombination operator (edrx by Whitley ([Wh-89])):-
From the two parents, we construct a table that regroups for each gene his adjacent neighbors. For example, the gene 1 in parent 1 has two neighbors the genes 2 and 7, and in the parent 2 the genes 5 and 3.

		*					
Parent1:	1	2	3	4	5	6	7
Parent2:	7	5	1	3	2	6	4

Table of neighbors	**1**	**2**	**3**	**4**	**5**	**6**	**7**
	2	1	2	3	4	5	6
	7	3	4	5	6	7	1
	5	6	1	6	7	2	4
	3			7	1	4	5

$\Longrightarrow$

child1 :	2	3	1	5	4	7	6
child2:	5	1	3	2	6	4	7

Table3:-Edge recombination operator (edrx).
To get the child1, we choose the first gene of the child1 randomly from the parent 1. His successor will be selected from the table of neighbors of the first gene with respecting the following rules:

The gene that will be elected must have the smallest number of distinct neighbors.
If we find several genes that verify the condition above, then we will choose between them one randomly.

The elected gene is placed in the child1 and is removed immediately from the table of neighbors. We restart until the complete construction of the child1.

Operator of maximal preservation crossover (MPX by Muhlenbein and al. ([Muh])):-
We choose two positions XP1 and XP2 at random. We look for the position of the gene (G.XP1.1) in parent2, we note it P1. Then, we place in the child1 from the P1 position the sequence of the genes between XP1 and XP2 of the parent1. The remainder of the child1 is copied from the parent2 with starting at his first gene and jumping all gene already been put in the child1.

		*		*			
Parent1:	1	2	3	4	5	6	7
Parent2:	7	5	1	3	2	6	4

child1 :	7	5	1	6	2	3	4
child2:	2	4	6	7	5	1	3

Table4:- Operator of maximal preservation crossover.

Order1 crossover operator (order 1 by Davis and al. ([Davis])):-
We choose randomly two positions XP1 and XP2 from parent1. The middle part is copied into the child1. The remainder is filled from (XP2+1) gene of the parent2 with jumping all elements that are already present in the child1.

		*		*			
Parent1:	1	2	3	4	5	6	7
Parent2:	7	5	1	3	2	6	4

child1 :	1	2	3	4	6	7	5
child2:	4	5	1	3	6	7	2

Table5:- Order1 crossover operator.

Order2 crossover operator (order 2 by Syswerda ([Syswerda])):-
We fill the child1 from the parent1 and we choose four points at random that we fill from their order of apparition in parent2. To build the child2, we make the same thing but with inverting the roles of parent1 and parent2.

		*		*		*	*
Parent1:	1	2	3	4	5	6	7
Parent2:	7	5	1	3	2	6	4

child1 :	1	7	3	2	5	6	4
child2:	7	3	1	4	2	5	6

Table6:- Operator of crossover order2.

Operator of position crossover (position by Syswerda ([Syswerda])):
To construct the child1, we choose four positions randomly that we are going to fill from the parent1 and the remainder of child1 is copied from the parent2 with starting since the beginning and jumping the genes that have already been put in child1.

		*		*		*	*
Parent1:	1	2	3	4	5	6	7
Parent2:	7	5	1	3	2	6	4

child1 :	5	2	1	4	3	6	7
child2:	1	5	2	3	7	6	4

Table7:- Operator of position crossover.

Operator of partial matched crossover (pmx by Goldberg and al. ([Gol-85])):-
We choose randomly two points of crossover XP1 and XP2. That decomposes each parent in three sequences.

S1	S2	S3

Table8:- A parent's partition.
The S1 and S3 sequences of the parent1 are copied into the child1. The S2 sequence of the parent2 is copied into the child1 with jumping the genes that has already been installed.

		XP1		XP2			
Parent1:	1	2	3	4	5	6	7
Parent2:	7	5	1	3	2	6	4

$\Longrightarrow$

child1 :	1	3	2	4	5	6	7
child2:	7	3	5	1	2	6	4

Table9:- Operator of partial matched crossover (pmx).
Uniform crossover operator (uox):-
The child1 is constituted with alternating randomly between the two parents.

Operators of mutation:-
The operators of mutation avoid establishing uniform population incapable to evolve. They consist in modifying the values of the genes of chromosomes. In opposition to the operators of crossover, they used only one parent to produce a unique child.

We use the following five operators of mutation.

Twors mutation:-
We exchange two gene positions chosen randomly in the parent.

		*		*			
Parent:	1	2	3	4	5	6	7

$\Longrightarrow$

child :	1	4	3	2	5	6	7

Table10:- Operator of twors mutation (twors).

Inverse center mutation (cim):-
We choose randomly one point of cut of the parent in two sequences S_1 and S_2. We take the first sequence S_1 and we reverse the order of the genes: the last becomes first, the next-to-last becomes second and so forth. We proceed in the same way on S_2.

				*			
Parent:	1	2	3	4	5	6	7

$\Longrightarrow$

child :	4	3	2	1	7	6	5

Table11:- Operator of inverse center mutation (icm).

Inverse mutation (im):-
We take a sequence S limited by two positions i < l chosen randomly. The order of the genes in the sequence S will be reversed by the same manner that what has been made on S_1 in cim.

		*			*		
Parent:	1	2	3	4	5	6	7

$\Longrightarrow$

child :	1	5	4	3	2	6	7

Table12:- Operator of inverse mutation (im).

Throas mutation:-
We construct a sequence of three genes of which the first is chosen at random and the two others are only these two successors. We proceed with the following manner: the first becomes second, the ancient second becomes last and the ancient last becomes first.

			*	→			
Parent:	1	2	3	4	5	6	7

$\Longrightarrow$

child :	1	2	5	3	4	6	7

Table13:- Operator of throas mutation (throas).

Thrors mutation:-
We choose three genes randomly that take the necessarily successive distinct positions $i < j < l$ while proceeding thus: the gene of the position i becomes to the position j and the one that was to this position will take lth position and the gene that occupied this position becomes to ith position.

		*		*		*	
Parent:	1	2	3	4	5	6	7

$\Longrightarrow$

child :	1	6	3	2	5	4	7

Table14:- Operator of thrors mutation (thrors).

Method of insertion:-
We used the method of insertion elitism that consists in recopying the best chromosome of the former population in the new. This will be supplement by the solutions results of operator of crossover and mutation. We respect the size of the population stay constant of generation in generation.

Numerical results:-
The operators of the genetic algorithm and their different modes, that will be used thereafter, are regrouped in this table:

Operator of crossover	Order1; Order2; position; cyclic; pmx; uox; MPX; edrx
Operator of mutation	cim; throas; im; Thrors; twors
Probability of crossover	1; 0.8; 0.6; 0.4; 0.3; 0.2; 0.1; 0
Probability of mutation	0; 0.1; 0.2; 0.4; 0.5; 0.6; 0.8; 0.9

Table15:- The different levels of the different operators.

We will work on the following benchmarks of traveling salesman problem :

Benchmark	size	known optimal value f_{opt}
Berlin52	52	7542
Eil101	101	629
KroA200	200	29368

Table16:- The T.S.P. benchmarks ([tsplib]).

These problems belong to the family of the symmetrical problems of T.S.P. Each is constituted of coordinates of points in Euclidian plan. We change at a time that only one parameter and we fix the other. We execute thirty times the genetic algorithm.

The program is writing in C language on PC machine of CPU 1.99Ghz. This machine used the SUSE Linux operating system.

In the beginning, we fixed the size of the population for the genetic algorithm at 104 for the different experiences on the three problems of T.S.P.. In order to discover the parameters; that it is the operator of crossover or the operator of mutation with their respective probabilities, that come out of the share and permits to get the best results.
We compare the found realizations in relation to Root Mean Square Error criterion:

$$\mathrm{RMSE} = \sqrt{\sum_{i=1}^{30} \frac{(f_i - f_{opt})^2}{30}}. \quad \text{Equ. (1)}$$

This criterion is equal to this formula:

$$\sqrt{\sum_{i=1}^{30} \frac{(f_i - f_{mean})^2}{30} + (f_{mean} - f_{opt})^2} \quad \text{Equ. (2)}$$

Obviously, minimizing the criterion Equ. (2) is equivalent to minimizing on the first hand the standard deviation and in the second hand the square of the mean of the gap.

Comparison between the operators of crossover:-
We set the probability of crossover at 1 and the probability of mutation at 0 and we get the following figure:

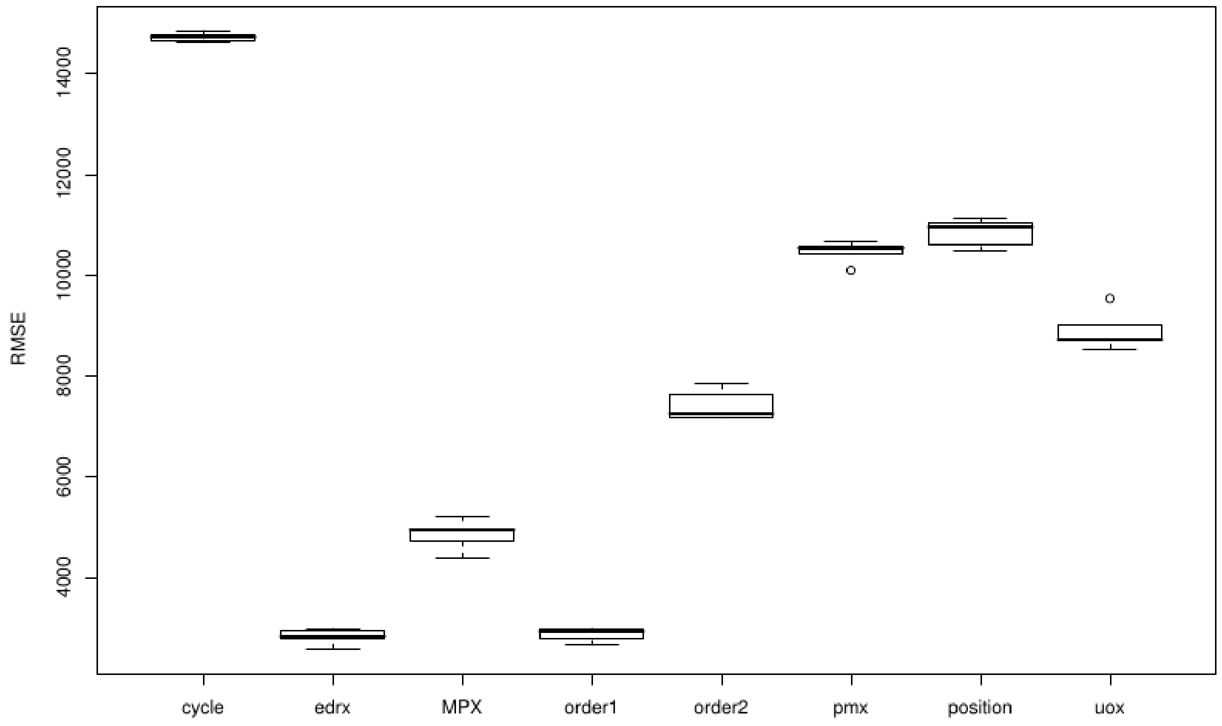


Figure2:- The boxplot of RMSE is according to the modification of crossover operator for berlin52.

From this figure, the best operator of crossover is **edrx** following by **order1**, **MPX**, **order2**, **uox**, **pmx**, **position**, and the worst is **cyclic**.

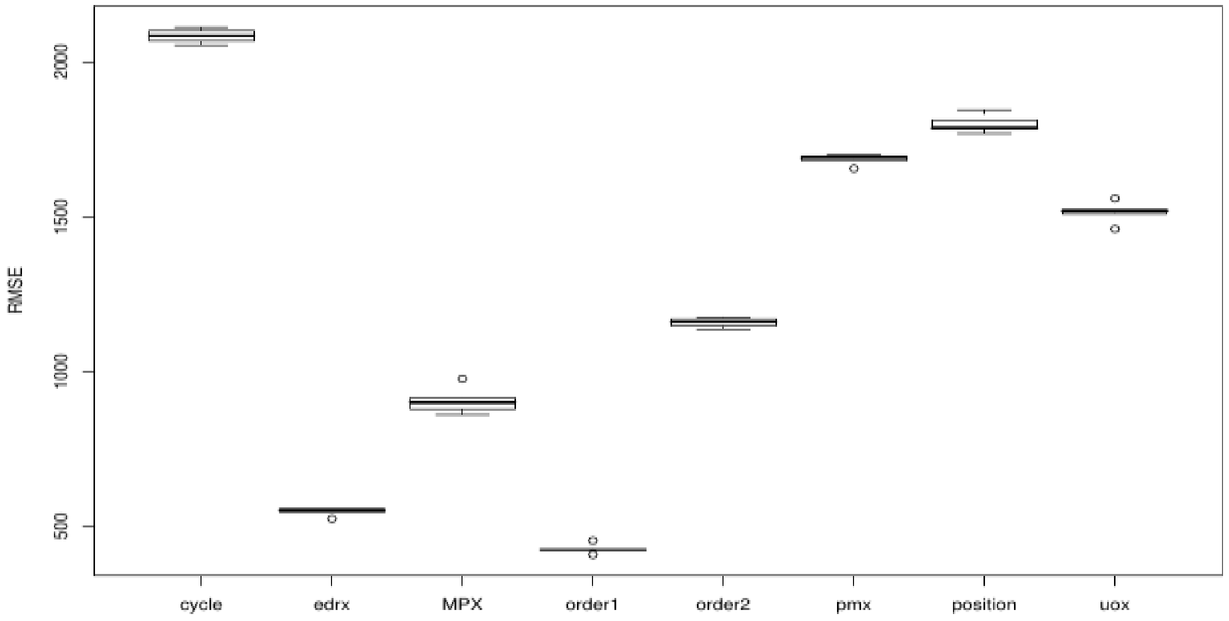


Figure3:- The boxplot of RMSE is according to the modification of crossover operator for eil101.

Corresponding to eil101 benchmark, we find that the worst operator of crossover is **cyclic** but the best is **order1** following by **edrx**.

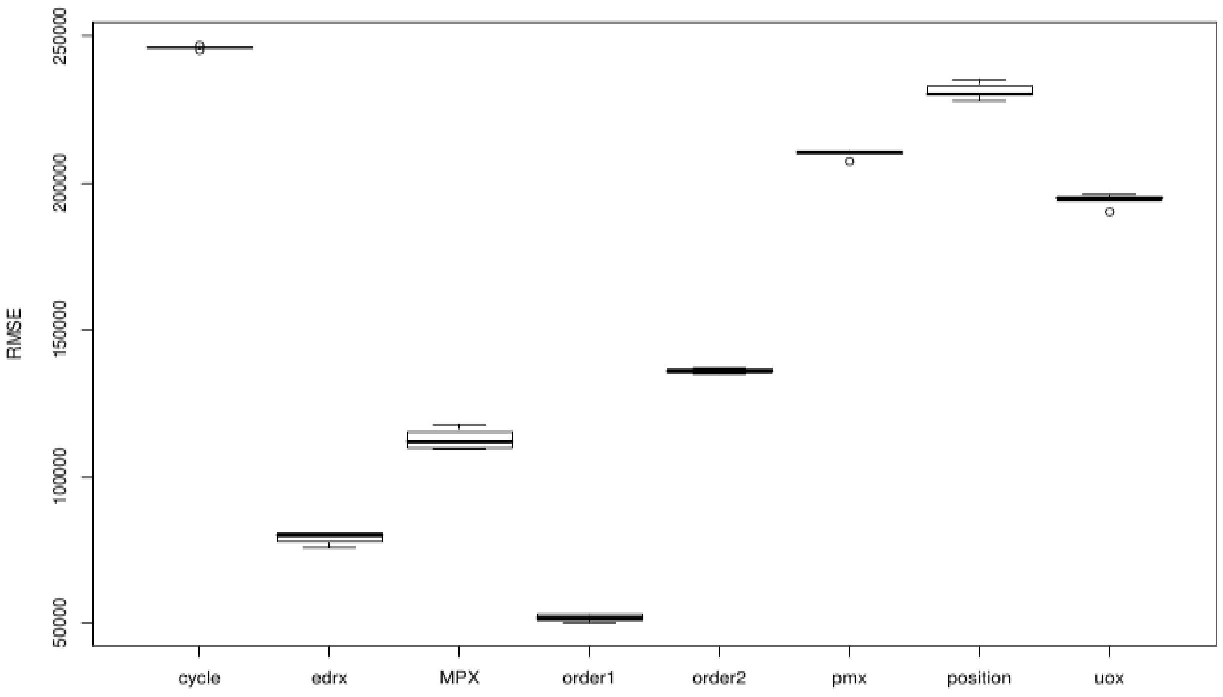


Figure4:- The boxplot of RMSE is according to the modification of crossover operator for kroA200.

As like as the results for the Eil101 problem, the operator of crossover **order1** shows his supremacy front of the **edrx** crossover operator.

From three benchmarks, we found in the first position the order1 operator of crossover following by **edrx** and in the third position **MPX**, after we found **order2**, **uox**, **pmx**, position and the worst is the **cycle** operator of crossover.

Comparison between the operators of mutation:-

We set the probability of crossover at 0 and the probability of mutation at 1. We find in the first position the **im** mutation following by **cim** mutation, **twors** in the third position. **Twors** is in the forth position and in the last position **throas**.

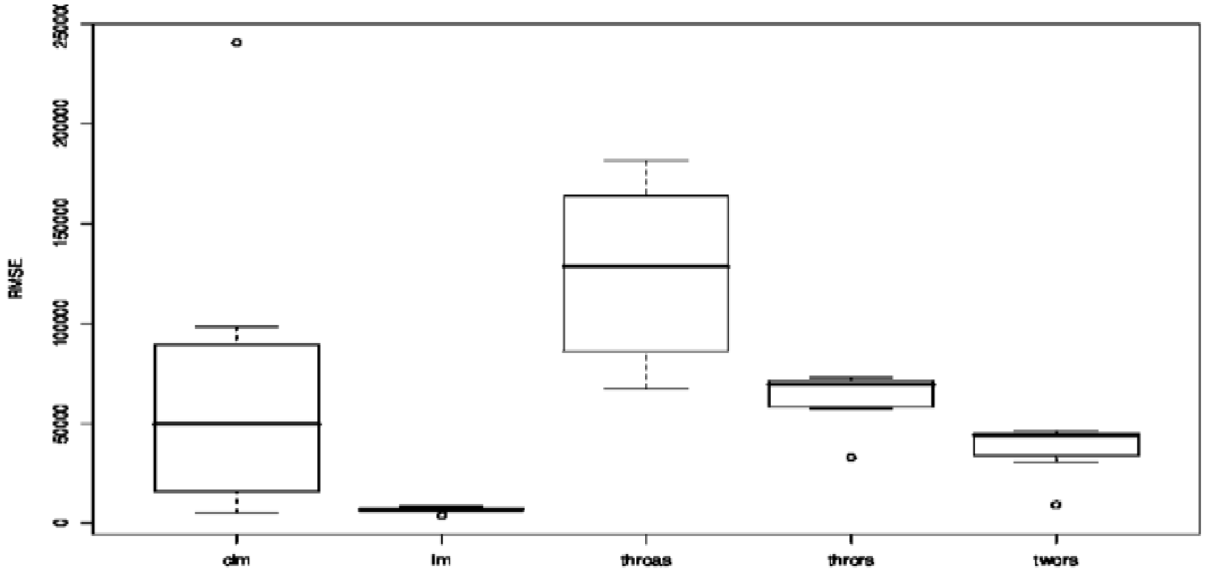


Figure5:- The boxplot of RMSE according to the modification of mutation operator for berlin52, eil101 and kroA200.

We find the same rank from three benchmarks. In conclusion, the best operator of mutation is **im**.

Study of interaction between operators of mutation and crossover:-

	berlin52					eil101					kroa200				
Rank	Cros.	P_x	Mut.	P_m	RMSE	Cros.	P_x	Mut.	P_m	RMSE	Cros.	P_x	Mut.	P_m	RMSE
1	edrx	0.4	im	0.9	834.74	edrx	0	im	0.9	78.93	edrx	1	im	0.9	3682.68
2	edrx	0	im	0.8	43.86	edrx	0	im	0.8	81.57	edrx	0	im	0.9	3836.76
3	MPX	0	cim	0.5	44.68	edrx	1	im	0.9	82.09	edrx	0.4	im	0.9	3919.81
4	MPX	0.2	im	0.9	845.59	edrx	0.4	im	0.9	82.15	edrx	0.2	im	0.9	4035.31
5	edrx	0.7	im	0.9	846.82	edrx	0.7	im	0.9	83.42	edrx	0.7	im	0.9	4090.82
6	edrx	0.4	im	0.8	50.31	edrx	0.6	im	0.9	83.76	edrx	0.8	im	0.9	4147.89
7	edrx	0.6	im	0.9	854.76	edrx	0.2	im	0.9	83.92	edrx	0.2	im	0.8	4203.11
8	edrx	0	im	0.9	857.01	edrx	0.2	im	0.9	84.28	edrx	0.6	im	0.9	4286.19
9	edrx	0.2	im	0.9	857.37	edrx	0.8	im	0.9	85.37	edrx	0	im	0.8	4357.87
10	edrx	1	im	0.9	67.89	edrx	.2	im	0.8	86.23	edrx	1	im	0.8	4417.76

Table17:-The ten best levels of operators

We observe that the operator of mutation which gives the smallest value of RMSE is **im** with **P_m = 0.9** for the three benchmarks. But, for the probability of crossover, we find three values **0.4, 0** and **1** for the same operator of crossover **edrx**.

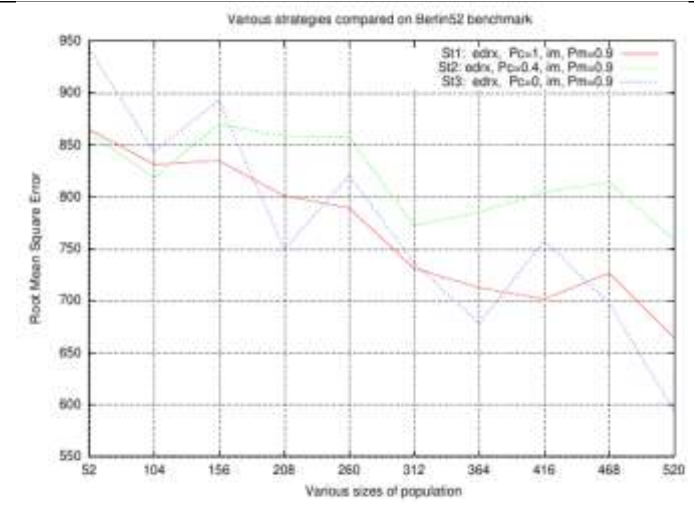
Size of population:-

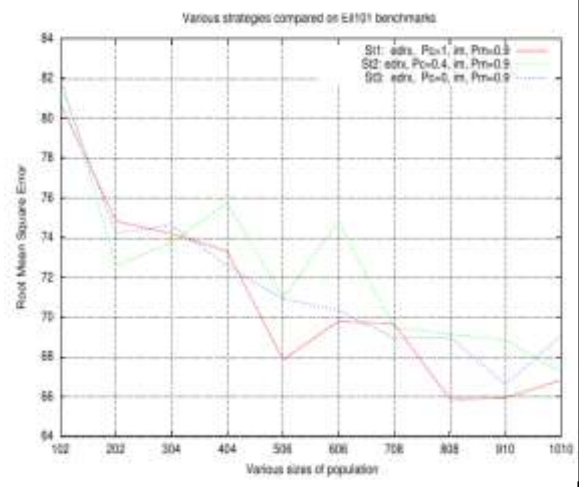
Until now, we have worked with fixed size of population at 104. We found three set of parameters that give the less value of the RMSE criterion:

1. **edrx**, P_x= 1 , **im** and P_m=0.9 called St1;
2. **edrx**, P_x= 0.4 , **im** and P_m=0.9 noted St2;
3. **edrx**, P_x= 0 , **im** and P_m=0.9 called St3;

But, what's the optimal size to give to genetic algorithm the manner to converge very closer to the optimum as possible?

To reply to this question, we fixed at each time the crossover and mutation operator and their probabilities as writing above and we tune the size of population from V to 10 × V for our genetic algorithm on the three benchmarks (with V is the size of the T.S.P. benchmark).





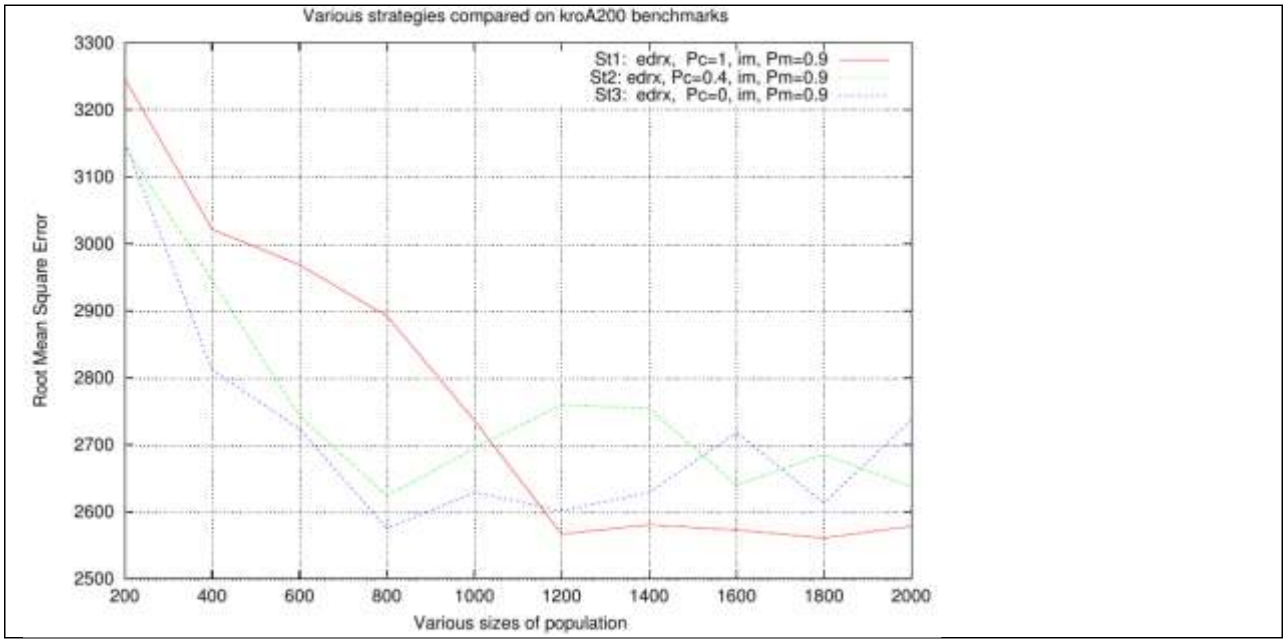


Figure6:- RMSE is according to the modification of the size of the population for berlin52, eil101 and kroA200.

From the picture of Berlin52, we observe that St3 give the minimum value of the RMSE criterion followed by St1. This strategy gives the values under the others curves from 6× V for eil101 benchmarks and from 8 × V for kroA200 benchmark and still until 10 × V for both benchmarks.

The St1 corresponding to **edrx** as crossover operator applied with probability equal to 1 and **im** as operator of mutation executed with probability equal at 0.9. This strategy is the better strategy with the size of population equal at 8 × V.

Conclusion:-

We used the genetic algorithm explained in the figure1. We displayed eight operators of crossover and five operators of mutation from the literature. In the beginning, we compared between the operators of crossover and we found this rank from the better to the bad: **order1**, **edrx**, **MPX**, **order2**, **uox**, **pmx**, **Position** and the last is **cycle**.
After, we observe that **im** is the best operator of mutation minimizing the **RMSE** criterion with taking the three T.S.P. benchmarks. Following by **cim**, **twors**, **thrors** and the bad operator is **throars**. When, we take the both operators: crossover and mutation. We have the stability in the operator of crossover **edrx**, operator of mutation (**im**) and his probability of mutation **0.9** but we have three values of probability of crossover (**1**, **0.4** and **0**).
To decide between them, we change the value of the size of population from **V** to **10 × V**. We find that the probability of crossover equal to 1 is better with the size of population equal at 8 **× V**.

References:-

1. **[Davis]** L. Davis. Applying Adaptive Algorithms to Epistatic Domains. In Proc. International Joint Conference on Artificial Intelligence, 1985.
2. **[Dav-93]** L. Davis, D. Orvosh, A. Cox and Y. Qiu. A Genetic Algorithm for Survivable Network Design. ICGA 1993: 408-415.
3. **[Gol-85]** D. Goldberg and R. Lingle. Alleles, loci, and the Traveling Salesman Problem. In Proc. International Conference on Genetic Algorithms and their Applications, 1985.
4. **[Gol-89]** D. Goldberg. Genetic Algorithm in Search, Optimization, and Machine Learning. Addison Wesley, 1989.
5. **[Hol-75]** J. H. Holland. Adaptation in Natural and Artificial Systems. The University of Michigan, 1975.
6. **[Mah-92]** S.W. Mahfoud and D.E. Goldberg. A Genetic Algorithm for Parallel Simulated Annealing. eds.), Parallel Problem Solving from Nature 2, Elsevier, 1992, 301-310.
7. **[Mah-95]** S.W. Mahfoud and D.E. Goldberg. Parallel recombinative simulated annealing: a genetic algorithm. Parallel Computing 21, 1995, 1-28.

8. **[Mich-96]** Z. Michalewicz. Genetic Algorithms + Data Structures = Evolution Programs. Springer-Verlag, 1996.
9. **[Mich-2000]** Z. Michalewicz and D.B. Fogel. \newblock{\em How to solve it: Modern heuristics. Springer-Verlag, 2000.
10. **[Muh]** H. Muhlenbein, How genetic algorithms really work: Mutation and hillclimbing. Parallel Problem Solving from Nature II, 15, 1992.
11. **[Oli-87]** I. Oliver, D. Smith, and J. Holland. A Study of Permutation Crossover Operators on the Traveling Salesman Problem. In Proc. Second International Conference on Genetic Algorithms and their Applications, 1987.
12. **[Syswerda]** G. Syswerda. Schedule Optimization Using Genetic Algorithms. In Handbook of Genetic Algorithms. l. Davis, ed. Van Nostrand Reinhold, New York, 1990.
13. **[Wh-89]** D. Whitley, T. Starkweather, and D. Fuquay. Scheduling Problems and Traveling Salesman: The Genetic Edge Recombination Operator. In Proc. Third Int'l. Conference on Genetic Algorithms and their Applications. J. D. Shaeffered. Morgan Kaufmann, 1989.