GENETIC ALGORITHM FOR OPTIMIZING DISTRIBUTION WITH ROUTE RESTRICTION CONSTRAINT DUE TO TRAFFIC JAMS

Nabil MOUTTAKI[1], BENHRA Jamal[2], Ghita RGUIGA[3]

1,2,3, OSIL Team, Lab LRI, ENSEM, UH2C, Casablanca, Morocco. - (nabil.mouttaki, jamal.benhra, ghita.rguiga)@ensem.ac.ma

KEYWORDS: Genetic algorithm; Crossover; Mutation; Initialization, Traveling salesman problems; constraints; traffic jams.

Abstract: The Travelling Salesman Problem (TSP) is a classical problem in combinatorial optimization that consists of finding the shortest tour through all cities such that the salesman visits each city only one time and returns to the starting city. Genetic algorithm is one of the powerful ways to solve problems of traveling salesman problem TSP. The current genetic algorithm aims to take in consideration the constraints happening during the execution of genetic algorithm, such as traffic jams when solving TSP. This program has two important contributions. First one is proposing simple method into taking in consideration an inconvenient route linked to traffic jams. The second one is the use of closeness strategy during the initialization step, which can accelerate the execution time of the algorithm.
The results of the experiments show that the improved algorithm works better than some other algorithms. The conclusion ends the analysis with recommendations and future works.

I. INTRODUCTION

Genetic algorithms (GA) are famous for their performances and their effectiveness to solve combinatorial optimization problems. Thanks to their efficiency, GAs have been applied to a wide range of fields such as medicine [1], software testing [2], industry [3,4], construction [5] or transport [6,7]. In all optimization problems, GA compete with other metaheuristics techniques such as particle swarm optimization, simulated annealing, evolutionary programming, tabu search, and ant colony optimization ACO. in their research work, researchers opt for the combination between different techniques or to do improvement inside one technique.

In the TSP problems, several heuristic and metaheuristic have been introduced since its definition. The first researcher to apply genetic algorithm to the TSP was Brady (1985) [8]. He was followed by Grefenstette et al. (1985) [9]; Goldberg and Lingle (1985) [10]; Oliver et al. (1987) [11] and many others. Plenty of researches provided responses to many defined problems but none to our knowledge has treated the restriction constraint due to traffic jams. In this work, we have developed an optimization algorithm for finding the shortest possible trip by taking in consider traffic jam.

This paper is structured as follow: chapter 2 synthesizes the state of the art and the principles in use in genetic algorithms, chapter 3 describes our approach, chapter 4 presents the experimentations and the results, and the final chapter gives a conclusion to this work.

II. STATE OF THE ARTS

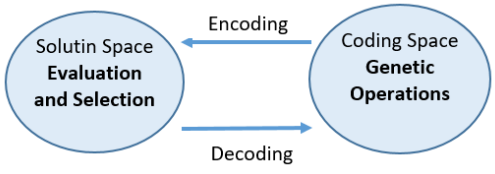
GA reproduce natural evolution mechanisms expressed by Darwin's principal of survival of the fittest. The figure 1 summarizes the mechanisms used to generate new candidates from an existing population until the finding of optimal solutions.

Start Genetic Algorithm
Initialize population;
Evaluate population;
While : not **Termination**
do Create new solutions:
Apply **Crossover** operator;
Apply **Mutation** operator;
Apply **Insertion**;
Evaluate new populations;
End While;
Decode chromosome

Fig1: Genetic algorithm mechanism

2.1 Encoding and Decoding of TSP

Genetic Algorithm works on two types of spaces at the same times, coding space (genotype) and solution space (phenotype). The phenotype describes the external appearance of an individual. The relation between phenotype and genotype is expressed as presented below:



In TSP, the most used type of encoding is Permutation Encoding [12] where the string of number represents the sequence of cities visited by the salesman. This encoding is also used for others ordering problem such as scheduling.

2.2 Initialization

Random initialization is the most used method to encode genetic algorithm for TSP application [9,10]. Consequently, the fitness function of the initial population has random results which will affect the performance of the next steps and therefore the whole algorithm performance [13,14].

Other methods were proposed to set the first generation, for examples:
a- The first individual is generated randomly. This individual will be mutated N-1 times.
b- The first individual is generated by using a heuristic mechanism such as the nearest neighbor.
This individual will be mutated N-1 times.
c- All first generation is generated by the same heuristic mechanism.

2.3 Fitness Function:

In TSP, each chromosome is a solution for the problem. The fitness of the chromosome is the sum of the distance between two consecutive cities. This distance is evaluated by the Euclidean Distance formula where the distance between two cities A(x1, y2) and B(x2,y2) is calculated as bellow:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

2.4 Management of constraints:

Genetic algorithms (GAs) are usually applied to unconstrained optimizations. When they are designed to deal with constraints many techniques have been developed. A simple technique is to attribute a constant penalty to the chromosomes that violate these constraints.

Other methods have been produced to allow GA's to manage constraints [15,16,17]:
a- Infeasible solutions are rejected after generation.
b- An invalid solution is approximated by its nearest valid one, or repaired to become a valid one
c- Special operators are designed in order to produce only feasible solutions.
d- The search space is restricted, and infeasible solutions are eliminated before chromosomes generations
All these techniques are time consuming and are difficult to build.

2.5 Selection Operator

The objective of this step is to select the suitable individual for matting [18,19,20].
Selection algorithm describes the methodology to choose parents for a meeting pool. These parents will create children for the next generation.
Four strategies are resumed here: roulette, rank, elite and tour.
a- In roulette selection, chromosomes are represented in the roulette wheel proportionally to their fitness functions.
b- In Rank selection, chromosomes are ranked in ascending order which is based on their fitness.
c- In elite strategy, best members are selected from the current population.
d- And in tournament selection, members are chosen, from the current generation, to compete and the best ones are selected to be a parent from the pool. This practice is continued until all members have been a part of competition.

2.6 Crossover Operator

Crossover operators play an important role in GA. Crossover technique is inspired from biology: children by inheriting their parents' genes can be more capable and may have better fitness than their parent. This concept was used in schema theory by Holland under the concept of building blocks.
Several crossover techniques [21] are created to reach in minimum iterations the optimum solution. In this paragraph, three crossovers are described: Partially mapped crossover (PMX), Cycle Crossover (CX) and Order Crossover Operator (OX).

Cycle Crossover Operator (CX)

CX was introduced by Oliver and al. [22]. This method identifies several cycles: Child 1 will be formed following cycles: cycle one is copied from the first parent, cycle two from the second parent, cycle three from the first parent, and so forth.

Child 2 will follow the same cycles, but the start will be from parent 2.
This example illustrates the process. The first position is chosen randomly from the first parent. In this case, it's the third position.

The Cycle 1 will be: 3 ➔ 1 ➔ 7.

We start with 3 and drop down to 1. 1 is found in the first position in Parent 1 and we drop down to 7. 7 drops down to 3 – The first cycle is finished.
As a result, the child inherits theses values and their positions form parent 1 with the remnants called from parent2 as seen in fig.

Cycle 2: 2 ➔ 4 ➔ 5 ➔ 6

We start with 2 and drop down to 4. 4 is found in the fourth position in Parent 1 and we drop down to 5. 5 Drops down to 6 and 6 drops down to 2 – The cycle is finished.

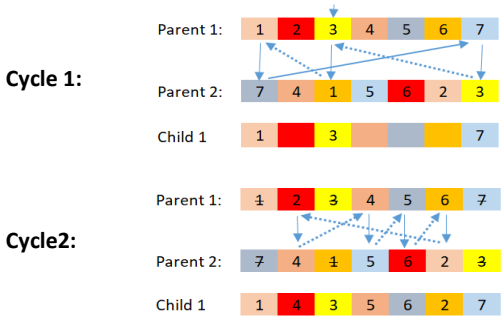


Fig 2. The Process of Cycle Crossover (CX)

Partially mapped Crossover Operator (PMX). PMX was introduced by Goldberg et al. [23]. This operator follows these steps:
a - Select randomly a subgroup from each parent:
For example,
P1 =A B C D E F G and P2 =E D F G B A C
b- Exchange subgroup between these parents
P1 =A B F G B A G and P2 =E D C D E F C
c - Map the relationship
d – Use this mapping map to legalize offspring

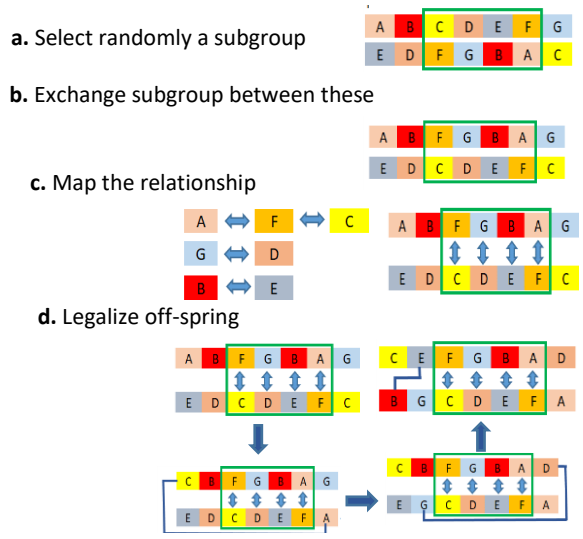


Fig 3. The Process of Partially mapped

Order Crossover Operator (OX). OX was proposed by Davis [24]. This process follows these steps:
a. Choice randomly a **subgroup from** a parent
b. Create a proto child by inserting the subgroup into the corresponding position of it. c. Delete the cities which are already in the subgroup from the 2nd parent. The remained cities will be used to fulfill the emptied cases in the child.
d. Place the cities into the emptied positions of the proto child from left to right following the the same order.

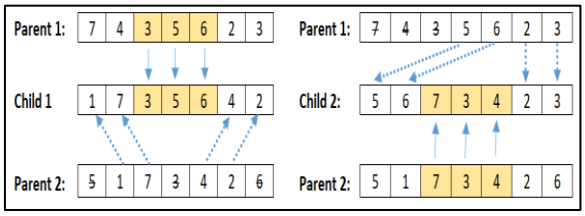


Fig4. The process of Order Crossover

2.7 Mutation Operator:

The main role of mutation step is to maintain diversity inside the population. Mutation operators [25] change the order inside one individual which helps to explore a new space and to avoid local minima's issue. In this paragraph, many mutation operators are described:

Reverse Sequence Mutation (RSM). It consists of inversing the order of cities inside one segment in the chromosome. The start and the end of this segment are chosen randomly.

Fig 5. Revers Sequence Mutation (RSM)

Insert Mutation. To perform Insert mutation, pick randomly two positions. Then move the second position to follow the first, shifting the rest along to accommodate.

Fig6. Insert Mutation

Inversion Mutation. First, pick randomly two positions and invert the substring between them.

Fig7. Inversion Mutation

Swap Mutation. In Swap mutation, two positions are randomly selected, and they swap their positions.

Fig8. Swap Mutation

Interchanging Mutation. Two random positions of the string are chosen and the positions corresponding to those positions are interchanged.

2.8 Insertion Operator:

The elite of the last generation are kept in the pool and will be inserted in the next generation without having undergone through the previous step.

2.9 Termination Operator:

Termination is a major part for the determination of an appropriate point in time to terminate the search. There are three popular termination strategies:

- Termination after a fixed number of generations
- Termination until solution meets the pre-set minimum requirement termination after reaching a plateau with no better results can be produced [26].

III. OUR APPROACH

In this chapter, we describe the main steps used in this algorithm to deal with traffic jam constraints.

3.1 Constraints:

Traffic jam is considered through the integration of constraints during fitness calculation. In this step, penalty is added to the cost of the distance between cities where traffic jam happened. The value of the penalty is important in order that this path will be eliminated during evaluation and future selection.
This choice is done to simplify the algorithm and to reduce the execution time. In fact, penalty is easily applied to all genetic operators without any change to do.

3.2 Encoding and Decoding of TSP

A real integer coding method is used to encode chromosomes. Each chromosome consists of two parts: the first part describes cities path and the second part is dedicated to containing chromosome information's. These information's are used for statistical uses.

3.3 Initialization

This work implements a simple method to generate individuals with better quality in the initial population: diversity and possible solution. It consists of adding constraints during the random choice: The choice of the first city to visit is made randomly among all cities. On the other hand, the choice of the next city to visit will be made randomly among the closest remaining cities to the current cities rather than all remaining cities. This option will be fixed through proximity parameter, which is defined before the execution of the algorithm, thus making it possible to adapt it to each treated case.

3.4 Selection Operator

In this work, individuals' selection is based on their fitness value. If an individual has lowest fitness value, he has more chance to be selected into the matting pool. The selection process combines different methods: Population classification and a random choice through each class.

During classification individuals are distributed: Winners, Elites, and the rest of the team. The winners are the three individuals with 3 best fitness value, the elites are the individual with top 10% value, the first league individuals are the one who has fitness value between 10 and 50%. The second league are the one remaining in the bottom.

The elite of the last generation are kept in the pool and will be inserted in the next generation without having to undergo the previous steps.

3.5 Crossover Operator

During crossover, pairs of parents are selected from the mating population. These parents will exchange their gene according to crossover rules: The crossover happens with a defined probability Pc. If it is applied, genes will be exchanged between the two parents to create new children who inherit their properties. If it is not applied the two parents are transferred to next step without any change.

3.6 Mutation Operator:

Reverse section mutation operator will be applied to the chromosomes. The chromosome will be chosen with probability Pm. If it's applied, genes will be reorganized as described in RSM operator above.

3.7 Insertion Operator:

The elite of the last generation are kept in the pool and will be inserted in the next generation without having undergone through the previous step.

IV. RESULTS AND DISCUSSION

MATLAB is used for programming. The program is tested in two times.

Here are the details from the implementation:

(1) Population size = 100.
(2) Selection ratio = 0.6.
(3) Mutation ratio = 0.05.
(4) Closeness = 10 cities
(5) Iteration max = 1000
(6) Traffic jam= 12-16-38

4.1 Initialization

To test initialization step, the executions were applied, with iteration max equal to zero, to three benchmarking instances: A280, Eil 76 and Berlin52. They were also applied to a real-world problem.

Case: A280

The choice for A280 was done for its 280 cities and its complexity. Figure9 presents an example of finding just after the first iteration. The results show that the closeness gives better results in the first instance. In the table1, the value with closeness option is better than with random initialization.

Average with closeness=5 : 6374.3
Average with closeness=10 : 8841.6
Average random : 31627.6

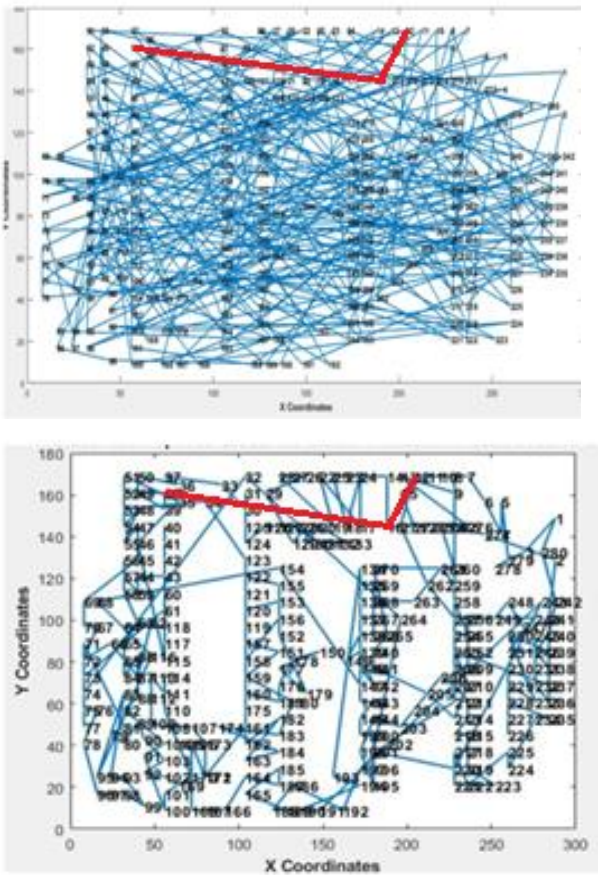


Fig 9. A280 path after iteration number 1: With random choice, with closeness parameters at 10

	A280		
Test N°	Closeness : 5	Closeness : 10	Random
1	6250,5	8681,4	31399,7
2	6454,9	8961,3	31702,3
3	6345,6	8710,5	31458,1
4	6253,5	8865,9	31241,9
5	6315,3	9022,8	31986,6
6	6458,3	8826,3	31308,8
7	6452,1	8752,2	31288,2
8	6278,7	8947,3	31875,1
9	6475,7	8906,6	32162,6
10	6458,6	8741,5	31852,7
Average	6374,3	8841,6	31627,6
Minimum	6250,5	8681,4	31241,9
Maximum	6475,7	9022,8	32162,6

Table1. Summary of simulation results for the effectiveness of initialization A280

Case: Eil 76

The same as for Eil76, the algorithm gives better result at the first iteration. For example, with closeness option at 5 the fitness function is 45% of the fitness function with a random initialization.

Average with closeness=5 : 1023.7
Average with closeness=10 : 1307
Average random : 2269.6

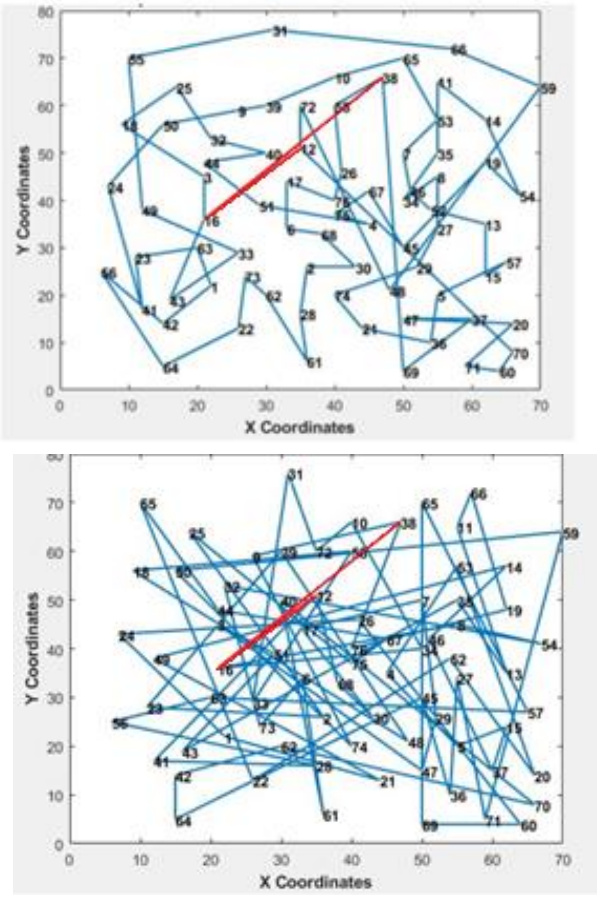


Fig 10. Eil76 path after iteration number 1: With random choice, with closeness parameters at 10

	Eil76		
Test N°	Closeness : 5	Closeness : 10	RANDOM
1	1029,0	1297,6	2284,1
2	1028,8	1350,6	2327,8
3	1012,6	1324,9	2319,0
4	1065,6	1283,6	2283,7
5	951,3	1282,6	2258,7
6	1021,9	1335,4	2214,1
7	1047,0	1284,3	2294,2
8	1018,3	1301,4	2232,8
9	1036,3	1295,9	2261,6
10	1026,5	1313,7	2219,5
Average	1023,7	1307,0	2269,6
Minimum	951,3	1282,6	2214,1
Maximum	1065,6	1350,6	2327,8

Table2: Summary of simulation results for the effectiveness of initialization Eil76

Case: Berlin52

The algorithm with closeness option provides better results just after the first iteration. The results in Table3 shows that the average with closeness option at 5 is 14291 so it's 44% less than the average with random initialization.

Average with closeness=5 : 14291
Average with closeness=10 : 18497
Average random : 25461

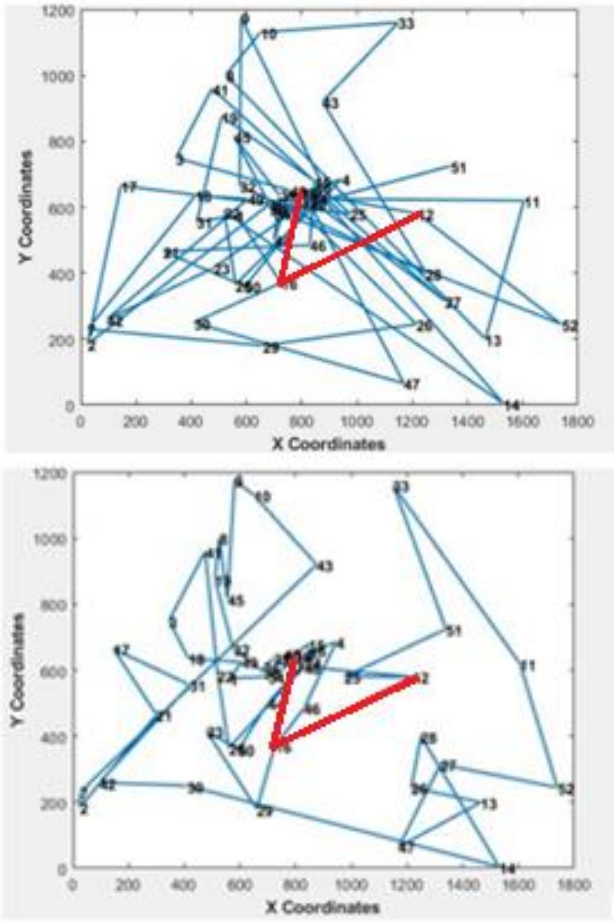


Fig 11. Berlin52 path after iteration number 1 : With random choice, with closeness parameters at 10

	Berlin 52		
Test N°	Closeness : 5	Closeness : 10	RANDOM
1	13762,9	18238,7	25518,3
2	14845,9	19217,0	24467,2
3	15081,1	18941,0	25819,4
4	14398,1	18698,3	24985,3
5	14158,3	18733,3	25186,5
6	13524,8	18151,7	25966,8
7	14219,6	18467,8	26507,9
8	14229,2	18753,6	26450,7
9	14586,5	17453,6	23615,2
10	14108,7	18324,5	26092,0
Average	14291,5	18497,9	25460,9
Minimum	13524,8	17453,6	23615,2
Maximum	15081,1	19217,0	26507,9

Table3: Summary of simulation results for the effectiveness of initialization Berlin52

This contribution has been peer-reviewed.
https://doi.org/10.5194/isprs-archives-XLIV-4-W3-2020-295-2020 | © Authors 2020. CC BY 4.0 License.

4.2 Restriction constraints

Restriction constraints were tested on the same benchmark instances. During the execution, we have changed the restricted paths 20 times to see if the restricted path were proposed as a best chromosome in any iteration.

Case: A280

The best value obtained is 3001.39. During the execution in any case the restricted path was confused with the best tour proposed. Also, the execution was similar to the case without restriction proving that penalty technique keeps the algorithm performance stable.

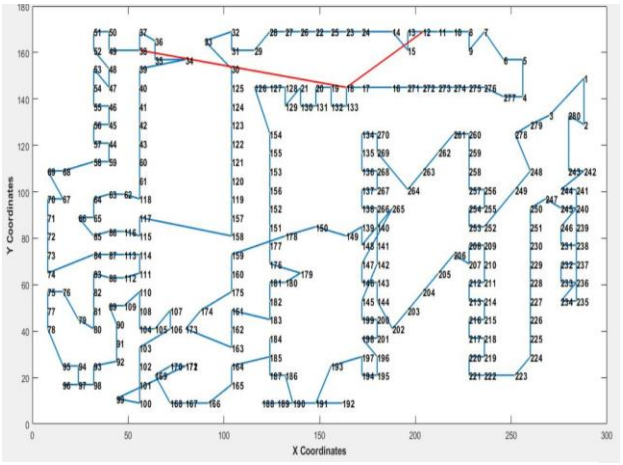


Fig 12. A280 path with constraints cities 12,18 and

Case: EiL76

EiL76 is well-known for its instance. Figure 13 presents an example of finding. During the execution, the restricted paths were separated from the current tour.

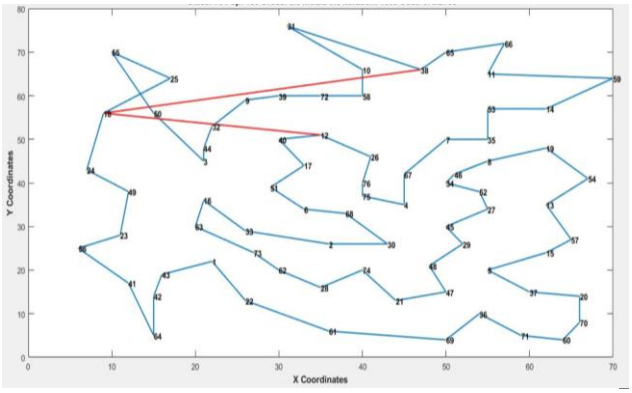


Fig 13. Eil76 path with constraints cities 12,18 and 38

Case: Berlin52

Berlin52 optimal tour is 7544. The algorithm maintains the same results as with other instances. Fig 14 shows an example of results obtained. The restricted route is colored in red and the path found is in blue. The restricted has never been confused with the path proposed during all the execution.

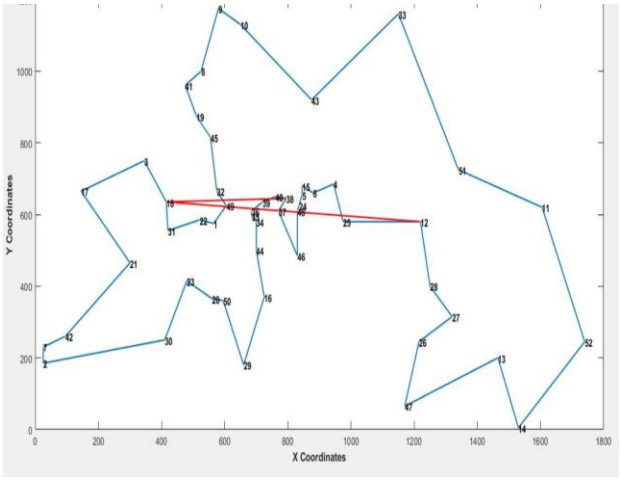


Fig 14. Berlin52 path with constraints cities 12,18 and 38

Case: Casa52

The algorithm was also tested on real case in Casablanca. Fifty-two distribution points were chosen in many areas in the cities. The algorithms were tested to check the possibility to respect the restricted paths.

Fig 15. A map visualization of the tour returned by the GA with constraints

V. CONCLUSION:

In this paper, we studied the effect of three techniques in the performance of genetic algorithms. The considered techniques were the nearest neighbor through the closeness option during the initialization, the elitism during the selection and the constrains when a path is restricted due to traffic jam.

We noticed that the closeness option improved the quality of the first generation which impacted the execution of the algorithm. We also noticed that the use of penalties simplifies the programming. In fact, penalty make the program independent from the genetic algorithm operators.
For future research, our algorithm can be improved to produce more strong solutions, which means that the restriction can take in consideration other constraints such as time window, the footprint or salesman capacity.

5. References

[1]. Ghaheri A., Shoar S., Naderan M. and Hoseini S.S. The applications of genetic algorithms in medicine. Oman Med. J., 30, 406–416 (2015).

[2] V. Sangeetha, T. Ramasundaram. "Application of Genetic Algorithms in Software Testing Techniques" International Journal of Advanced Research in Computer and Communication Engineering Vol. 5, Issue 10, (October 2016).

[3] L. García-Hernández, A. Araúzo-Azofra, L. Salas-Morera. "A Review on Encoding Schemes used by Genetic Algorithms in Plant Layout Design", XIII CONGRESO INTERNACIONAL DE INGENIERÍA DE PROYECTOS Badajoz, (8-10 de julio de 2009).

[4] L. Davis, "Job shop scheduling with genetic algorithms," in Proceedings of the 1st International Conference on Genetic Algorithms, pp. 136–140, Lawrence Erlbaum Associates, Pittsburgh, Pa, USA, (1985).

[5] AL-TABTABAI, H. and ALEX, A.P., "Using genetic algorithms to solve optimization problems in construction", Engineering, Construction and Architectural Management, Vol. 6 No. 2, pp. 121-132 (1999).

[6] H. El Hassani, J. Benhra, and S. Benkachcha, "Utilisation des algorithmes génétiques (AG) dans l'Optimisation multi-objectif en logistique avec prise en compte de l'aspect environnemental (émissions du CO2)," in Colloque international LOGISTIQUA, RABAT, (2012).

[7] A.H. Sabry, J.Benhra, and H.El hassani, A Performance Comparison of GA and ACO Applied to TSP, International Journal of Computer Applications, Volume 117, N°19 (May 2015).

[8] RM Brady, "Optimization strategies gleaned from biological evolution," Nature, vol. 317, no. 6040, pp. 804 (1985).

[9] John Grefenstette, Rajeev Gopal, Brian Rosmaita, and Dirk Van Gucht, "Genetic algorithms for the traveling salesman problem," in Proceedings of the first International Conference on Genetic Algorithms and their Applications. Lawrence Erlbaum, New Jersey (160-168), (1985).

[10] David E. Goldberg and Robert Lingle, "Alleleslociand the traveling salesman problem," in ICGA, (1985).

[11] IM Oliver, DJ Smith, and JRC Holland, "A study of permutation crossover operators on the tsp, genetic algorithms and their applications," in Proceedings of the Second International Conference on Genetic Algorithms, pp. 224–230, (1987).

[12] Larrañaga, P., Kuijpers, C., Murga, R. et al. Genetic Algorithms for the Travelling Salesman Problem: A Review of Representations and Operators. Artificial Intelligence Review 13, 129–170 (1999).

[13] A. B. Hassanat, V. B. S. Prasath, M. A. Abbadi, S. A. Abu-Qdari, H. Faris, An improved Genetic Algorithm with a new initialization mechanism based on Regression techniques. Information (Switzerland). 9, doi:10.3390/info9070167 (2018).

[14] P. V. Paul, P. Dhavachelvan and R. Baskaran, "A novel population initialization technique for Genetic Algorithm," 2013 International Conference on Circuits, Power and Computing Technologies (ICCPCT), Nagercoil, pp. 1235-1238 (2013).

[15] C.A. Coello Coello, Theoretical and numerical constraint-handling techniques uses with evolutionary algorithms: a survey of the state of the art, Comput. Meth. Appl. Mech. Eng. 191 1245–1287, (2002).

[16] Z. Michalewicz, M. Schoenauer, Evolutionary algorithms for constrained parameters optimization problems, Evol. Comput. 4 1–32 (1996).

[17] Alice E. Smith and David W. Coit "Constraint-Handling Techniques - Penalty Functions,", in Handbook of Evolutionary Computation, Institute of Physics Publishing and Oxford University Press, Bristol, U.K., Chapter C5.2. (1997).

[18] Paul, P. Victer, P. Dhavachelvan and R. Baskaran. "A novel population initialization technique for Genetic Algorithm." 2013 International Conference on Circuits, Power and Computing Technologies (ICCPCT): 1235-1238 (2013).

[19] Jadaan, Omar Al, Lakishmi Rajamani and C. Raghavendra Rao. "IMPROVED SELECTION OPERATOR FOR GA 1." (2008).

[20] J. Zhong, X. Hu, M. Gu, J. Zhang, "Comparison of Performance between Different Selection Strategies on Simple Genetic Algorithms," Proceeding of the International Conference on Computational Intelligence for Modelling, Control and automation, and International Conference of Intelligent Agents, Web Technologies and Internet Commerce, (2005).

[21] A.J. Umbarkar and P.D. Sheth," CROSSOVER OPERATORS IN GENETIC ALGORITHMS: A REVIEW", ICTACT JOURNAL ON SOFT COMPUTING, VOLUME: 06, (OCTOBER 2015).

[22] I. M. Oliver, D. J. Smith, and J. R. C. Holland, "A study of permutation crossover operators on the TSP", Proceedings of the 2 nd International Conference on Genetic Algorithms on Genetic Algorithms and their Application, pp. 224-230, (1987).

[23] David E. Goldberg and Robert Lingle Jr., "Alleles, loci and the traveling salesman problem", Proceedings of the 1st International Conference on Genetic Algorithms, pp. 154- 159, (1985).

[24] Lawrence Davis, "Applying adaptive algorithms to epistatic domains", Proceedings of the 9th international joint conference on Artificial Intelligence, Vol. 1, pp. 162- 164, (1985).

[25] N. Soni, and T. Kumar, "Study of Various Mutation Operators in Genetic Algorithms" (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 5 (3), 4519-4521, (2014).

[26] Abdel-Rahman Hedar, Bun Theang Ong, and Masao Fukushima "Genetic Algorithms with Automatic Accelerated Termination "Technical report, Dept. of Applied Mathematics and Physics, Kyoto University, (2007).