Optimizing Genetic Algorithms for the Multiple Traveling Salesman Problem: A Random-Mixed Mutation Approach

Eva Cantín Larumbe, Eva Teisberg, Mikel Baraza Vidal, Francesco Pio Capoccello, Daniele Borghesi
Bachelor's degree in Data Science
Universitat Politècnica de València
Valencia, Spain

***Abstract*—This paper delves into the resolution of the Multiple Travelling Salesman Problem (MTSP) through the application of a Genetic Algorithm. Our research encompasses a systematic exploration of numerous combinations of selection methods, crossover operators, mutation operators, and population replacement strategies. Through extensive experimentation, we meticulously tested hundreds of configurations to identify the most effective combination. Subsequently, the optimal algorithm underwent an additional layer of refinement with the exploration of hundreds of hyperparameter combinations.**

The key innovation lies in the utilization of a unique form of mixed random mutation, harnessing the synergies of several mutation operators well-established in the literature. The resulting algorithm demonstrated superior performance across diverse MTSP instances.

Furthermore, our findings highlight the significance of certain parameter settings for optimal algorithmic performance. Notably, a high crossover rate and a high mutation rate were identified as beneficial, along with the utilization of very large initial populations.

***Index Terms*—Genetic Algorithm, TSP, MTSP, Multiple Traveling Salesman Problem**

I. Introduction

The Multiple Travelling Salesman Problem (MTSP) is an extension of the well-known Travelling Salesman Problem (TSP) by enabling the utilization of multiple salesmen (vehicles) in its resolution.

At any given time, a specific group of cities is visited by only one salesman, ensuring that each location is visited exactly once. Additionally, each salesman's journey starts and ends at the same position (a depot), creating distinct tours for each vehicle. The primary objective of MTSP includes minimizing the total distance incurred by the tours undertaken by the salesmen.

MTSP finds extensive applications across various real-world scenarios in business, industry, and engineering. It holds significant practical importance for organizations involved in logistics, transportation, and scheduling. Effectively solving the MTSP can lead to substantial cost savings, improved operational efficiency, and enhanced resource utilization.

In addressing the multiple traveling salesman problem (MTSP), various techniques, including heuristics, metaheuristics, and mathematical optimization methods, have been explored in the literature. Our focus lies on proposing a Genetic Algorithm-based solution, a result of extensive experimentation involving hundreds of combinations of operators and hyperparameters.

Specifically, we introduced a mutation technique that exploits the actions of well-established and widely used mutation operators. By blending the characteristics of multiple mutation operators, our approach aims to strike a balance between exploration and exploitation in the solution space, enhancing the algorithm's capability to discover high-quality solutions.

The paper is organized through the following sections:

- II - Related Work: we reviewed existing literature related to the mTSP problem and its resolution.
- III - Algorithm design: we delve into the core of our approach. We detail the algorithm implemented and used to solve the mTSP problem.
- IV - Experiments: we present the results of experiments based on several tests of hundreds of operators and hyperparameters.
- V - Conclusions: the final section summarizes the findings, discusses the practical implications, and outlines potential future directions.

II. Related work

Singh et al. (2009) in [1] present an alternative solution to the current problem using a steady-state grouping Genetic Algorithm. In contrast to our chromosome phenotype representation, they opt not to store the salesperson associated with each route, instead only storing the route itself. In addition to minimizing the total distance covered by each salesman, they introduce a fitness function aimed at minimizing the maximum distance traveled by any one salesperson. This approach seeks to balance the workload among salesmen. Although we also explore the Tournament Selection method, it is not ultimately incorporated into our final Genetic Algorithm, unlike in their work. Furthermore, their choice of crossover and mutation operators differs from those proposed in this paper. In terms of population generation, they implement a replacement strategy that prioritizes the best fitness values, replacing children with higher fitness values to the detriment of the "worst" ones. Additionally, their population generation relies on randomness, ensuring only the uniqueness of population members. Regarding their experiments, they primarily focus on two aspects:

obtaining results with different chromosome representations proposed in the literature and evaluating their own approach. They conduct comparisons based on the total distances covered by all salesmen and execution times, as well as the total distance covered by each salesman in conjunction with execution times.

The problem addressed by Bolaños et al. (2015) in [2], offers a more complex variant of the problem, the Multi-Objective Multiple Travelling Salesman Problem (MOMTSP). Here, the objective is twofold: to minimise both the total distance travelled and to balance the salesmen's working time. To achieve these goals, the authors implement the non-dominated sorting Genetic Algorithm (NSGA-II), integrating it with a local search strategy.The algorithm operates on an initial population of randomly generated solutions, subsequently creating a new generation through selection, recombination and mutation. Solutions are evaluated according to the concept of Pareto dominance. In this paper, the authors focus on practical tests, using real instances to evaluate the performance of their algorithm in realistic transport and logistics scenarios. The first scenario involves a Colombian courier company in Cali with 29 customer locations (nodes) and a central depot. The challenge is to efficiently plan routes for three salesmen (m=3) responsible for visiting these nodes. In the second instance, inspired by existing literature, there are 75 customers and one depot. Travel times are simulated realistically by randomly generating speeds between 20 km/hour and 90 km/hour for each connection between nodes. Like the first scenario, the goal is to optimize routes for three salesmen (m=3) to enhance the efficiency of the delivery process.

Zhou et al. (2018) in [3] propose two Partheno Genetic Algorithms (PGA) for this purpose. One PGA employs roulette selection and elitist selection, while the other, named IPGA, integrates selection and mutation. The authors critique the original PGA's genetic operations, highlighting issues with random mutation, parameter setting, and convergence ability, leading to the development of IPGA. The core innovation of the Improved Partheno Genetic Algorithm (IPGA) lies in its adoption of a novel random mixed mutation approach, which proves to be instrumental in achieving superior results. This strategic use of mixed mutations addresses shortcomings identified in the original Partheno Genetic Algorithm (PGA) by introducing a more diverse and effective set of mutation operations. The IPGA's unique approach involves randomly selecting individuals, creating temporary populations, and applying a mix of mutation operations, including Swap, Insertion, Inversion, Scramble, Modify Breaks, and combinations thereof. This method contributes significantly to the algorithm's improved performance, demonstrating its efficacy in overcoming issues related to genetic operations, convergence ability, and solution quality in solving complex instances of the Multiple Traveling Salesman Problem (MTSP). The IPGA's success underscores the importance of the random mixed mutation strategy in enhancing the algorithm's overall efficiency and effectiveness.

Wang et. al (2020) in [4] introduce an Improved Partheno-Genetic Algorithm with Reproduction Mechanism (RIPGA) for addressing the Multiple Travelling Salesman Problem (MTSP) with multiple depots and closed paths. The study explores various Genetic Algorithm configurations, including a two-part chromosome encoding, a reproduction mechanism inspired by Invasive Weed Optimization (IWO), a mixed selection operator, and group evolutionary processes (consisting of 7 mutation methods). This is to aim to overcome the communication limitations between the individuals in the population and improve the algorithm's ability to find high-quality solutions. Effective communication within the population is crucial for optimizing MTSP. Limited communication can hinder the exchange of important information about optimal routes, resulting in suboptimal solutions. Improved communication mechanisms ensure broader exploration of the solution space, facilitate the exchange of genetic material and prevent premature convergence towards suboptimal solutions. However, our scenario addresses a slightly different challenge, the problem of Multiple Travelling Salesman with a single depot. In this case, unlike Wang's multiple depot scenario, all vehicles (salesmen) must start and end their route at a single depot. This difference introduces unique constraints and optimisation objectives that we need to be aware of when potentially implementing configurations from Wang's study.

Xin-Ai Dou et al. (2020) in [5] present a comparative study of ten commonly used crossover operators in Genetic Algorithms for solving the traveling salesman problem (TSP). The study systematically compares these operators and conducts extensive experiments on TSP instances of varying sizes to investigate their optimization effectiveness. The results of the study identify the sequential constructive crossover (SCX) and the zoning crossover (ZX) as the two most effective operators for Genetic Algorithms to solve TSP. The paper also provides a guideline for choosing a suitable crossover operator for TSP problem-solving purposes. Moreover, this comparative study can have broader applications and serve as a reference for similar problems such as vehicle routing and route planning for unmanned aerial vehicles. It can also tackle the multiple postman problem effectively.

III. Algorithm design

In this section, all the components of the proposed Genetic Algorithm are detailed. Table I encapsulates a schematic representation of the parameter values and notation.

TABLE I
PARAMETERS SETTING OF THE PROPOSED ALGORITHM

Parameter	**Value**
Population size (μ)	*400*
Crossover rate (α)	*1.0*
Mutation rate (β)	*0.5*
Early-Stopping limit (γ)	*100*

A. Chromosome representation (genotype)

To facilitate the operation of crossover and mutation operators, phenotypes (solutions) are internally encoded within a

D	2	4	1	D	8	6	D	5	7	9	3

Fig. 1. An example of phenotype representation with 3 routes (salesmen).

unique list of n_L locations from 0 to $n_L - 1$. Location 0 (the *Depot*) is symbolized as D.

As can be seen from Figure 1, the number of Ds is always equal to the number of salesmen n_S (routes). This representation proves to be easy and straightforward to process, especially for crossover and mutation operators.

During the calculation of the fitness,[1] an additional D at the end of the list allows the fitness function to include the cost of the last vehicle's return to the depot, correctly calculating the quality of each solution.

The chosen representation has a drawback: crossover and mutation operators might create children with empty routes (more Ds consecutive). To avoid this issue, elements other than D are extracted from the original solution p and placed in the same order in a new temporary list p_{temp}. Then, the crossover operation (section III-E) or mutation operation (section III-D) are performed on the list p_{temp}. In the end, elements other than D within the original solution p are replaced as per the new order in p_{temp}. The new and updated p is used as output.

B. Population generation

To generate each member p of a population P of size μ, an algorithm with the following steps is employed:

1) Generate a solution in the form of an empty list of length equal to the number of locations n_L - 1 plus the number of salesmen n_S.
2) Initialize the first element of the list ($p[0]$) as D, representing the starting point of the first route (first salesman).
3) Choose a number of random positions within the list, equal to the number of remaining salesmen (n_s - 1). The positions are chosen from $\{2, 4, 6, ..., n-6, n-4, n-2\}$.
4) The positions generated in step 3 are initialized with D. Each position simultaneously represents the starting point of each route (salesman) and the endpoint of the previous route.
5) The remaining elements of the solution list are initialized with the locations (excluding location 0) in random order.

The methodology in step 3 avoids the positioning of consecutive D in the list, ensuring that empty routes cannot exist.

C. Selection method

The selection method involves selecting parental chromosomes for subsequent operations. The *Fitness-Proportional Selection* (*Roulette-Wheel*), picks λ chromosomes considering their selection probabilities derived from their fitness values:

[1]As is common in the MTSP, fitness is calculated as the total distance traveled by all salesmen.

1) For each solution p within the population P of size μ, calculate the probability ρ of being selected based on the fitness f_p (equation 1)

$$\rho_p = \frac{f_p}{\sum_{i=1}^{\mu} f_i} \tag{1}$$

2) Choose a random value λ less than μ. Then, for λ times:
   a) Generate a random value r between 0 and 1
   b) Select the first solution p in P that has a ρ less than r

D. Crossover operator

The crossover operator in a Genetic Algorithm combines genetic information from two parent solutions to generate offsprings, promoting diversity and exploration. The proposed operator is the *Cut-and-Crossfill Crossover*.

For each pair of solutions (p_1 and p_2) among those selected by the selection method, the operator generates a random value r_c between 0 and 1. If the random value r_c is greater than the crossover rate α, no crossover is performed, otherwise:

1) Select a random position i.
2) Generate two new solutions (children) c_1 and c_2, copying the first segment (up to i) of p_1 into c_1 and the first segment of p_2 into c_2.
3) Scan the values of p_1 starting from i, and insert in c_1 those not yet present. Perform the same procedure with p_2 and c_2.

This method effectively prevents duplicate numbers within an individual, ensuring compliance with the permutation representation. The Crossover Rate α affects the trade-off between exploration and exploitation. A higher Crossover Rate promotes exploration, introducing diverse genetic material from the parents. In contrast, a lower Crossover Rate favors exploitation by reducing the number of crossovers

E. Mutation operator

Mutation operators in a Genetic Algorithm introduce random changes to individual solutions, fostering exploration of the solution space. The proposed mutation operator exploits a hybrid system, partly inspired by the system proposed by Zhou et al. (2018) in [3] (see section II). As explained in the reference section, the main characteristic of the exposed strategy is to use multiple mutation operators simultaneously on different members of the population. For each child c created by the crossover operator, a random value r_m is generated between 0 and 1. If the random value r_m is greater than the mutation rate β, no mutation is performed on the child c, otherwise:

1) A mutation method between *Swap*, *Scramble*, *Insertion*, or *Inversion* is randomly chosen.
2) The chosen mutation operator is applied to the child c, generating c'.
3) c is replaced by c'.

The behavior of the mentioned mutation operators is briefly explained below. Given a chromosome p of length n, the gene

at position $i \in \{0, ..., n\}$ is selected, as well as the gene at position $j \in \{0, ..., i-1, i+1, ..., n\}$. Subsequently:

- **Swap Mutation:** the content present in $p[i]$ is exchanged with that present in $p[j]$.
- **Insertion Mutation:** the content of $p[j]$ is moved to $p[i+1]$, shifting all elements between i and j to the right.
- **Inversion Mutation:** the order of elements between i and j is reversed.
- **Scramble Mutation:** the order of elements between i and j is randomly shuffled.

These internal operators also prevent duplicate numbers within an individual, as the permutation representation requires. The mutation rate β acts similarly to what was illustrated for the crossover rate α (see section III-D).

F. Replacement strategy

Population replacement strategies in a Genetic Algorithm determine how new offspring solutions replace existing ones, shaping the evolution process. The chosen strategy for our GA is *Lambda-Mu Replacement*.

The *Lambda-Mu Replacement* uses the λ offspring created through crossover and mutation operations combined with the current population of μ solutions. The combined solutions are sorted based on their fitness values, and only the best μ individuals are selected to form the next generation.

The number of offspring λ (*Lambda*) and the dimension of the population μ (*Mu*) values can have an impact on the behavior of the algorithm. A higher λ value contributes to exploration by introducing more diverse genetic material, while a higher μ value emphasizes exploitation by focusing on selecting the most promising individuals.

G. Algorithm scheme

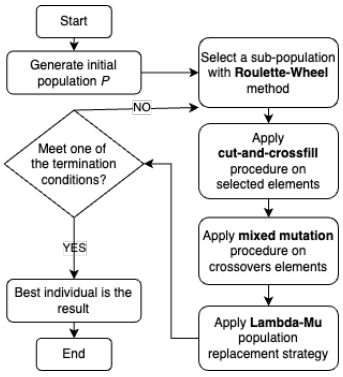

Fig. 2. Flowchart of the proposed Genetic Algorithm.

In conclusion, the proposed algorithm follows the scheme illustrated in Figure 2 until it encounters one of the termination conditions:

- The preset time limit for execution is reached.
- A number of generations equal to the early stopping limit γ is generated without any improvement in the quality (fitness) of the best solution.

IV. Experiments

As mentioned, the experimental phase of the work was divided into two distinct stages:

1) **Operators Grid-Search:** search for the best combination of operators (parents selection, crossover operator, mutation operator, population replacement strategy).
2) **Hyperparameters Grid-Search**: search for the best combination of hyperparameters for the best combination found in stage 1.

For each of the two phases, a Grid Search was performed among hundreds of combinations. Specifically, in stage 1 all the following operators were tested in various combinations:

- **Selection methods:** *Fitness Proportional Selection* (FPS), *Linear Ranking Selection* (LRS) with the parameter s equal to 1.5, *Exponential Ranking Selection* (ERS), with the parameter c equal to 0.5, *Tournament Selection* (TS) with the the tournament size equal to 0.3 (30%) and the tournament probability equal to 0.5, *Uniform Selection* (US).
- **Crossover operators** (with a *Crossover Rate* α equal to 0.8): *Cut-and-Crossfill Crossover* (CCX), *Partially Mapped Crossover* (PMX), *Edge Crossover* (EX), *Sequential Constructive Crossover*.
- **Mutation operators** (with a *Mutation Rate* β equal to 0.3): *Swap Mutation* (SWM), *Insertion Mutation (ISM), Inversion Mutation* (IVM), *Scramble Mutation* (SCM), *Random-Mixed Mutation* (RMM).
- **Population Replacement Strategies:** *Generational Replacement* (GR), *Replace Worst (genitor) Replacement* (RWR), *Round-Robin Replacement* (RRR), *Lambda-Mu Replacement* (LMR).

Once the best combination of operators was obtained from stage 1, the stage 2 was performed among the following hyperparameters combinations:

- **Population size:** 25, 50, 100, 200, and 400 solutions.
- **Crossover rate (α):** 0.6, 0.7, 0.8, 0.9, and 1.0.
- **Mutation rate (β):** 0.1, 0.2, 0.3, 0.4, and 0.5.
- **Early-Stopping limit (γ):** 25, 50, and 100 generations.

The majority of the tested operators are well-known and used in the literature. However, we decided to implement, in addition to RMM, the SCX crossover introduced by Ahmed et al. (2010) in [6].

The algorithm chooses the city stored in the first gene of parent 1 as the start city. Than, for each subsequent city to be visited, find the two cities, k and h, to the right of the last visited city in both parents:

- If both k and h are unvisited, compare their distances (d_{ih} and d_{ik}) to the last visited city (city i). Select k if $d_{pk} < d_{ph}$, otherwise select h.
- If one of k or h is visited, find the first unvisited city from the associated parent for comparison.
- If both k and h are visited, find the first unvisited city from each parent and select the one with the shortest distance to the last visited city as the next city to visit.

The algorithm repeats the above steps using parent 2 as the template to generate another offspring.

For both grid searches, each combination of operators was tested on 8 different instances, each with a varying number of locations and traveling salesman, along with a matrix indicating the distances between the various locations. The first row of the distance matrix (location 0) represents the depot: the starting and ending point for each traveling salesman. For each instance, the algorithm was executed with 3 different populations (runs) to maximize the reliability of the results. The time limit for the termination condition was set to 180 seconds.

To determine the ranking of combinations, a method inspired by that proposed by Derrac et al. (2011) in [7] was used. For each instance, the average fitness values obtained from different runs is calculated. This calculation is performed for each tested combination, and then a ranking of all of them is performed based on the average fitnesses. In the end, the average ranking and the median ranking among all the instance rankings is calculated. The best combination in both grid searches was chosen based on the average rank (lower is better). In case of equal values, the combination with the lower median rank would be selected.

400 combinations were tested in the first grid search, and 375 combinations in the second grid search, for a total of 775 experiments.

A. Solutions analysis

D	8	16	7	4	3	14	6	5	15	10	9	12	2	13	D	11	D	1
D	13	2	12	9	6	15	14	5	3	4	16	7	8	D	11	D	10	1
D	1	10	12	9	7	16	8	3	4	15	6	5	14	D	11	D	13	2

Fig. 3. Solutions for instance 1 with random seed 0, 1 and 2 (from the top)

The solutions of the algorithm vary greatly from instance to instance, and also based on the initialized random population (random seed). From figure 3 it can be observed that the algorithm tends not to create routes of uniform length. The best solutions exhibit a consistent pattern with one route significantly longer than the others. A similar behavior is also observed with other instances.

It can be hypothesized that in many instances, certain locations very close to the depot persist. In these cases, the algorithm prefers to use only one of the salesmen to cover the majority of distant locations and utilize the other salesmen for short trips to the nearby locations, thus minimizing the impact of the return to the depot.

B. Operators performance comparison

Table II illustrates that, among the top 5 GAs, the selection method stands out as the sole variable that differs in only 4 out of the 5 configurations. This suggests that the choice of selection method might not exert significant influence on the models' ranking. Conversely, it appears that factors like CCX, RMM, and LMR exert a more notable impact on the outcomes.

TABLE II
TOP 5 CONFIGURATIONS FROM THE FIRST GRID-SEARCH

Sel.	Cross.	Mut.	Repl.	Average Rank	Median Rank
RWS	CCX	RMM	LMR	**13.25**	10.50
US	CCX	RMM	LMR	14.63	9.50
US	PMX	SWM	LMR	14.75	14.00
LRS	CCX	RMM	LMR	18.88	**8.50**
ERS	CCX	RMM	LMR	19.50	13.00

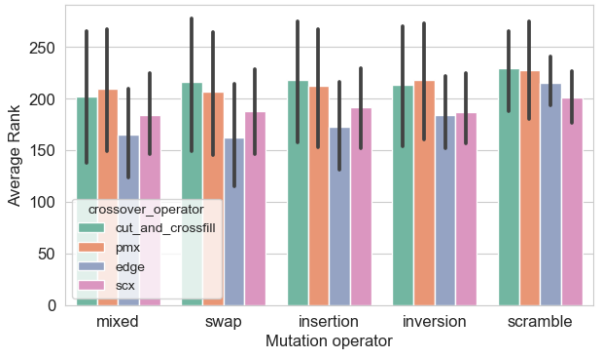

Fig. 4. Comparison of Average Ranks among Mutation Operators combined with Crossover Operators.

The insights obtained from Figure 4 underscore a pattern where the combination of mutation operators with the EX and SCX operators consistently yields a lower average rank compared to the other two crossover operators. However, the inclusion of the CCX operator in the most successful combination from the experiments, rather than EX or SCX, suggests that the effectiveness of crossovers is closely tied to all other operators, not just the mutation operator.

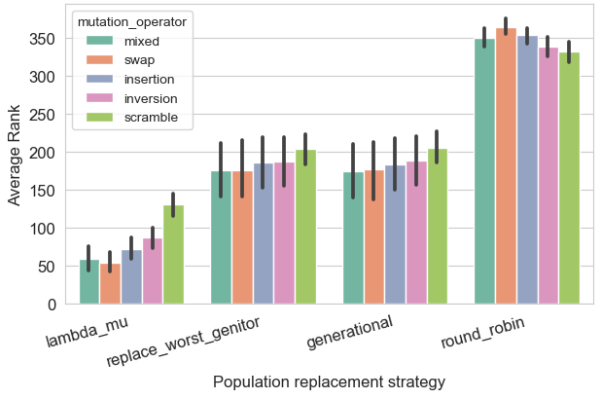

Fig. 5. Comparison of Average Ranks among Population replacement strategies combined with Mutation Operators.

Figure 5 clearly shows that LMR replacement strategy stands out for its remarkably robust behavior, consistently demonstrating very low average ranks across all tested mutation operators. This observation suggests that LMR might excel in swiftly converging towards high-quality solutions, irrespective of the mutation operator employed.

C. Hyperparameters influence comparison

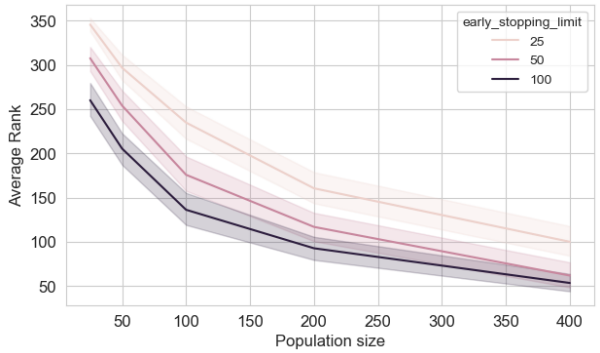

Fig. 6. Evolution of Average Rank with Population size and Early stopping limit.

In Figure 6, our results show that the performance of the algorithm consistently improves as the population size increases, peaking at a population size of 400 individuals. The increased performance with a larger population indicates that the algorithm benefits from a more comprehensive exploration of the solution space. At the same time, pushing the algorithm through a longer and extensive exploration of the solutions space with a high early stopping limit (100) showed to obtain the better ranks. The results highlight how an enhancement of the initial population's quality could be taken into consideration to improve the algorithm. The population generation function could therefore be modified with new strategies.

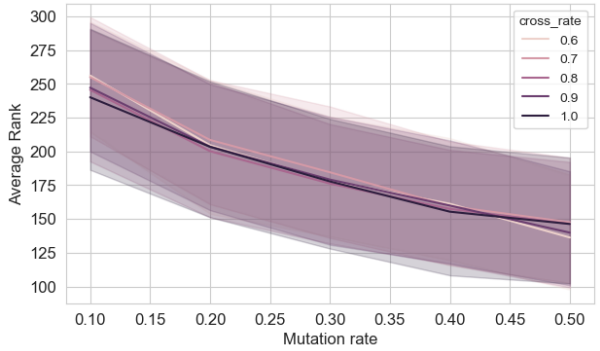

Fig. 7. Evolution of Average Rank with Mutation rate and Crossover rate.

In Figure 7, our comprehensive exploration of different mutation rates reveals substantial improvement in efficiency with higher mutation rates: a mutation rate of 0.5 results in a lower average rank than in the other cases. Instead, no particular differences across crossover rates are detected.

This observation suggests that a higher mutation rate contributes positively to the exploration of different genetic variations in the algorithm. The algorithm seems to require a high rate of variation and modification of the initial population, further supporting the hypothesis that an improvement in the generation of the initial population could lead to better results.

V. Conclusions

In this paper, we addressed the Multiple Travelling Salesman Problem (MTSP) through the application of a Genetic Algorithm. Our research involved extensive experimentation, testing hundreds of configurations to identify the most effective combination of selection methods, crossover operators, mutation operators, and population replacement strategies.

The key innovation in our approach lies in the utilization of a form of mixed random mutation, blending the characteristics of several well-established mutation operators in the literature. This hybrid mutation approach, combined with a Lambda-Mu population replacement strategy, demonstrated superior performance across diverse MTSP instances.

Furthermore, our findings highlighted the significance of certain hyperparameters for optimal algorithmic performance. Notably, a higher population size and a higher mutation rate were identified as highly beneficial, suggesting the importance of a diverse and explorative initial population in the Genetic Algorithm.

A. Future improvements

Future research can delve into the exploration and implementation of more sophisticated initialization strategies for the population. While the current study utilized a random generation approach, incorporating heuristic-based or problem-specific initialization methods may yield better starting solutions.

Furthermore, enhancing the algorithm's adaptability by exploring adaptive operator selection is another area for future improvement. The current study identified effective combinations of operators through a Grid Search. However, developing mechanisms for adaptive operator selection based on the problem instance's characteristics during the algorithm's runtime could lead to more flexible and efficient optimization.

Moreover, the exploration of higher values for key hyperparameters, particularly population size and mutation rate, could be beneficial. Investigating whether increasing the population size beyond 400 individuals or utilizing mutation rates greater than 0.5 leads to further improvements in algorithm performance merits attention.

References

[1] Singh, Alok, and Anurag Singh Baghel. "A new grouping Genetic Algorithm approach to the multiple traveling salesperson problem." Soft Computing 13 (2009): 95-101

[2] Bolaños, Rubén, Mauricio Echeverry, and John Escobar. "A multiobjective non-dominated sorting Genetic Algorithm (NSGA-II) for the Multiple Traveling Salesman Problem." Decision Science Letters 4.4 (2015): 559-568

[3] Zhou, Honglu, Mingli Song, and Witold Pedrycz. "A comparative study of improved GA and PSO in solving multiple traveling salesmen problem.", 2018

[4] Wang, Zijian, et al. "An improved partheno-Genetic Algorithm with reproduction mechanism for the multiple traveling salesperson problem.", 2020

[5] Dou, Xin-Ai, et al. "A Comparative Study on Crossover Operators of Genetic Algorithm for Traveling Salesman Problem.", 2023

[6] Ahmed, Zakir H. "Genetic algorithm for the traveling salesman problem using sequential constructive crossover operator.", 2010

[7] Derrac, Joaquín, et al. "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms.", 2011

B. Annex

Eva Cantín Larumbe reviewed [1]
Francesco Pio Capoccello reviewed [2]
Mikel Baraza Vidal reviewed [5]
Eva Teisberg reviewed [4]
Daniele Borghesi reviewed [3]