INTERNATIONAL JOURNAL ON INFORMATICS VISUALIZATION

journal homepage : www.joiv.org/index.php/joiv

Optimizing Genetic Algorithm by Implementation of An Enhanced Selection Operator

Mohammed BinJubier [a,b], Mohd Arfian Ismail [a,c,*], Muhaini Othman [d], Shahreen Kasim [d], Hidra Amnur [e]

[a] *Faculty of Computing, Universiti Malaysia Pahang Al-Sultan Abdullah, Pekan, Pahang, Malaysia*
[b] *Engineering Faculty, Sana'a Community College, Sana'a, Yemen*
[c] *Centre of Excellence for Artificial Intelligence & Data Science, Universiti, Al-Sultan Pahang, Lebuhraya Tun Razak, Gambang, Malaysia*
[d] *Faculty of Computer Science and Information Technology, Universiti Tun Hussein Onn Malaysia, Parit Raja, Johor, Malaysia*
[e] *Department of Information Technology, Politeknik Negeri Padang, Padang, Indonesia*

*Corresponding author: *arfian@umpsa.edu.my*

***Abstract*—The Traveling Salesman Problem (TSP) represents an extensively researched challenge in combinatorial optimization. Genetic Algorithms (GAs), recognized for their nature-inspired approach, stand as potent heuristics for resolving combinatorial optimization problems. Nevertheless, GA exhibits inherent deficiencies, notably premature convergence, which diminishes population diversity and consequential inefficiencies in computational processes. Such drawbacks may result in protracted operations and potential misallocation of computational resources, particularly when confronting intricate NP-hard optimization problems. To address these challenges, the current study underscores the pivotal role of the selection operator in ameliorating GA efficiency. The proposed methodology introduces a novel parameter operator within the Stochastic Universal Selection (SUS) framework, aimed at constricting the search space and optimizing genetic operators for parent selection. This innovative approach concentrates on selecting individuals based on their fitness scores, thereby mitigating challenges associated with population sorting and individual ranking while concurrently alleviating computational complexity. Experimental results robustly validate the efficacy of the proposed approach in enhancing both solution quality and computational efficiency, thereby positioning it as a noteworthy contribution to the domain of combinatorial optimization.**

***Keywords*— Genetic algorithm; traveling salesman problem; selection operator.**

Manuscript received 5 May 2024; revised 24 Aug. 2024; accepted 11 Sep. 2024. Date of publication 30 Nov. 2024.
International Journal on Informatics Visualization is licensed under a Creative Commons Attribution-Share Alike 4.0 International License.

I. INTRODUCTION

The Traveling Salesman Problem (TSP) is one of the extensively researched challenges in combinatorial optimization. The problem involves a given set of cities with known distances between them, aiming to determine the shortest path that visits each city exactly once and returns to the starting city. Despite being classified as NP-Hard, the constraint remains that the traveling salesman must traverse each city once and conclude the journey at the initial city [1], [2], [3].

Two predominant approaches exist for addressing the TSP. The first involves exact algorithms characterized by exhaustive search methods, providing a best solution. This approach strives for precision, albeit at the cost of computational intensity. The second method, known as heuristic algorithms or approximate methods, encompasses algorithms that yield optimized or specific solutions without guaranteeing optimality for all problem instances [3], [4]. Due to the computational demands associated with exact algorithms, heuristics algorithms, such as Genetic Algorithms (GA), become preferable as they offer insights into the magnitude of suboptimal solutions attainable within reasonable time frames [5], [6], [7], [8].

The GA is a widespread evolutionary method for efficiently working out the TSP [3], [5]. Rooted in the survival of the fittest principle, the GA generates superior solutions from a population of potential solutions for TSP. Initially, a random population of individuals is generated, with only the most suitable individuals surviving and reproducing. The GA iteratively creates new chromosomes using crossover and mutation operators until predefined conditions, such as convergence or a fixed time, are met. The objective is to uncover solutions with outstanding fitness values to pursue the best solution. A notable feature of the GA is its ability to

investigate the entire search space with the assistance of the whole population [9], [10]. An important observation is that good sub-solutions in a GA become increasingly similar as they evolve. Consequently, computations of good sub-solutions in later generations often duplicate earlier efforts. The challenge is to eradicate these redundant computations in the early generations of a GA, significantly reducing computation time while maintaining or improving the result's quality [10].

Hence, finding a trade-off between exploration (i.e., exploring promising areas of the search space) and exploitation (i.e., utilizing existing identified genetic operators to search for the optimum) is essential. Therefore, the performance of the GA is highly dependent on two factors [10], [11]. The first factor is selection, which is used to choose the set of chromosomes for the mating process, and the second factor includes genetic operators (crossover and mutation) employed to create new individuals and introduce random changes.

The main objective of this study is to enhance the selection operator, which plays a significant role in the GA process. A novel parameter in Stochastic Universal Selection (SUS) is proposed with the intention of effectively narrowing down the search space and exploiting the genetic operators of promising candidate solutions. The aim is to enhance result quality within a reasonable time frame. The study rigorously evaluates and compares the performance of the proposed method, the enhanced GA (EGA), against the traditional GA. Additionally, the EGA undergoes testing with various problem sizes from the TSPLIB dataset [12].

The manuscript is structured into four sections. Section 1 thoroughly reviews related works, critically examining the existing literature in the field. Section 2 furnishes an exhaustive description of the EGA methodology, encompassing a detailed exposition of the novel parameter operator. Section 3 presents experiments and their corresponding results, comprehensively analyzing the empirical findings. The concluding section, Section 4, encapsulates the study's termination, offering summative insights and proposing potential directions for future research endeavors.

Research in GAs focuses on reducing computation time and improving the worth of the final product. Improving the efficiency of GA is a crucial pursuit, particularly when addressing complex optimization problems. The selection operator is pivotal in this enhancement, serving as a vital exploratory tactic within the search space and guiding the algorithm toward convergence [11]. Moreover, genetic operators such as crossover and mutation significantly facilitate the introduction of random alterations and the generation of new individuals [10]. Numerous studies and applications of GAs, employing both sequential and parallel approaches, have been elucidated with the goal of diminishing the time required to identify a viable solution and augmenting the quality of the final product. Parallel GAs, hybrid GAs, and substantial modifications to the evolutionary process or GA design represent some strategies explored in this endeavor [13]. These methodologies aim to address the dual objectives of computational efficiency and optimization efficacy.

Zakir H. Ahmed [14] conducted an examination and comparison of three crossover types—Sequential Constructive Crossover (SCX), Generalized Npoint Crossover (GNX), and Edge Re-combination Crossover (ERX)—for solving the TSP. The empirical findings of this study demonstrate that SCX yields a higher quality solution than GNX and ERX. Aranganayaki [15] introduced a novel approach for solving the TSP using GA, enhancing the Sequential Constructive Crossover (SCX) method. This improved method efficiently generates a high-quality solution within a realistic timeframe by selecting the minor edge from the parent and developing a new child or retaining the same parent, contingent on the edge selection. In this study, the researcher endeavored to minimize runtime by executing this method once and utilizing SCX to achieve a high-quality solution. In addressing the TSP, Rani, and Kumar [16] employed the Roulette Wheel and Stochastic Universal Sampling methods, complemented by the Order Crossover (OX) technique. Notably, the Stochastic Universal Sampling method yielded satisfactory results when dealing with small population sizes. However, the GA demonstrated improved solution quality for larger population sizes by employing the Stochastic Universal Sampling method in conjunction with the Elitism Method.

Employing an efficient mutation strategy, K. Rani and V. Kumar [16] introduced an innovative approach to parent selection for the TSP. This method randomly selects two cities from the solution, followed by an iterative heuristic transpose procedure. Three operators, namely transpose, shift-and-insert, and swap, are utilized in the subsequent heuristic steps. The GA with Efficient Mutation (GA-EM) was then compared with four contemporary, efficient algorithms. The results demonstrated superior accuracy performance of GA-EM across all tested TSP instances.

Proposed by A. Hussain and Y. Muhammad [10], Split Rank Selection (SRS) introduces an alternative selection strategy that adeptly navigates the delicate balance between exploration and exploitation. This method not only resolves the fitness scaling issue but also maintains sufficient selection pressure throughout the entire selection process. Within this framework, individuals undergo ranking based on their fitness values, ranging from the worst to the best. Notably, even individuals with identical fitness values receive distinct rankings. SRS has exhibited commendable results, particularly in scenarios characterized by a limited population size. Phuang [17] proposed an algorithm for solving the symmetric TSP. The algorithm is based on the galaxy-based search algorithm and employs a novel technique known as the clockwise search process. This novel approach is intended to conduct a thorough search within promising areas. Furthermore, the algorithm uses the hill-climbing local search algorithm to improve neighbor search capabilities. Rigorous testing with relevant data sets demonstrates the proposed algorithm's efficacy. Gao [18] introduced an innovative algorithm that diverges from the traditional GA by utilizing regular random numbers for determining crossover and mutation locations. Drawing inspiration from the weed algorithm, this approach incorporates a typical distribution sequence as a library of random numbers. The exchange and different locations of genetic factors are retrieved from this library and applied iteratively to address the TSP. MATLAB tests demonstrate that the algorithm proposed in this study exhibits certain advantages over the conventional genetic

algorithm, particularly in path optimization and runtime efficiency.

Islam et al. [19] introduced a parallel GA for the TSP using the Map/Reduce Framework on a Hadoop cluster to enhance solution quality. Parallel GA was intended to reduce TSP runtime and expedite execution. This study incorporates three distinct crossover methods: Order Crossover, Two Point Crossover, and Partially Matched Crossover. The implementation of GA in parallel is facilitated through the Map/Reduce framework, allowing the algorithm to handle extensive datasets effectively. The storage of large datasets is managed using the Hadoop Distributed File System (HDFS).

II. Material and Methods

A. The Proposed Enhanced Genetic Algorithm

Broadly, the EGA proposed herein adopts a stochastic methodology, strategically exploring optimal solutions for the TSP within a specified timeframe. The principal objective lies in augmenting the GA's efficacy and elevating solution quality for TSP instances. The EGA delineated in this study encompasses four pivotal steps, elucidated in Fig 1. The ensuing sub-sections furnish a meticulous exposition of these four stages.

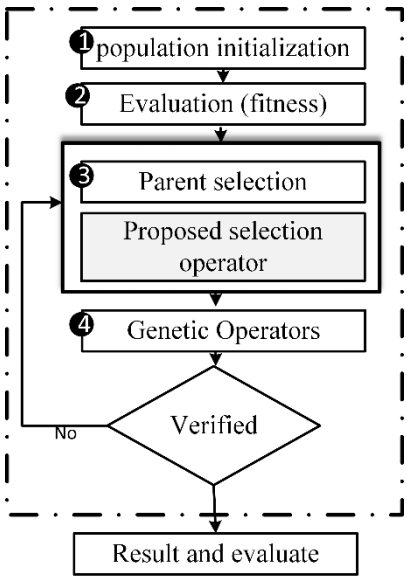

Fig. 1 The enhanced genetic algorithm steps

B. Population Initialization

The algorithm commences by randomly initializing the population, generating an initial state characterized by randomness that progressively evolves towards a more structured state, thereby rendering it well-suited for algorithmic evaluation. This population encompasses the potential solutions to a given problem, illustrated in the context of the traveling salesman problem, where each solution is denoted as a chromosome or an individual. Ensuring diversity within the population is imperative, with a stipulation that each member differs from others and that the population size is appropriately controlled. The length of an individual is typically determined by the number of nodes or genes in the problem, as illustrated in Fig. 2 [20].

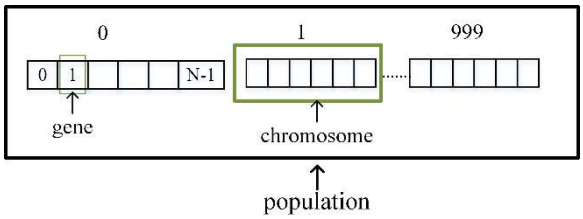

Fig. 2 A population of individuals

C. Evaluation

The evaluation function, commonly known as the fitness function, functions as a heuristic measure to estimate solution quality. In the fitness function phase, each individual solution generated in the preceding step is assigned a fitness value, with this value computed through techniques tailored to user specifications [21]. One widely adopted method is represented by Equation 1, designed to determine the minimum length of a Hamiltonian tour. In this context, a Hamiltonian tour is defined as a closed path systematically visiting each n node in a graph G exactly once. Therefore, the optimal solution to the TSP requires a permutation of the node indices 1,2,⋯,n ensuring the minimum length (〖length〗 _((min))). In this equation, d_((i)(i+1)) denotes the distance between nodes i and (i+1), while d_((i)(1)) signifies the distance between node n and the first node [22], [23], [24]

$$length_{(min)} = \sum_{i=1}^{n-1} d_{(i)(i+1)} + d_{(n)(1)} \quad (1)$$

D. Selection of New Candidate

The selection phase is pivotal in guiding the GA toward optimal solutions and effectively limiting the search space. Its objective is to leverage the favorable characteristics of promising candidate solutions, progressively enhancing their quality. Theoretically, this iterative process will lead the GA toward an appropriate and satisfactory solution for the optimization problem. Despite extensive research spanning years, there have yet to be universally applicable guidelines or robust theoretical foundations for selecting an optimal operator for each problem. This lack of guidance poses a significant challenge, as suboptimal operator selection can detrimentally impact both the speed and reliability of the GA's performance [25].

The proportionate selection operator, a prominent method in the selection phase, entails choosing individuals based on their fitness values. This operator can be implemented through diverse approaches, such as the Roulette Wheel and SUS. In contrast, the ordinal-based selection operator selects individuals based on their orderliness within the population. In ordinal selection, the selection pressure remains independent of fitness, assigning individuals with higher fitness to a greater probability of participating in the mating process for generating the next generation. Consequently, this mechanism enables the GA to focus on promising regions within the search space. Noteworthy methods falling under this category include the Exponential Rank Selection operator, Linear Ranking Selection operator, and Tournament Selection operator [25], [26], [27], [28].

1) Roulette wheel selection: Roulette Wheel Selection (RWS), also recognized as Fitness Proportional Selection and introduced by Holland in 1975, is a stochastic method widely employed in GAs for parent selection [26], [29]. This

mechanism selects individuals based on a probability directly proportional to their fitness scores.

In the context of RWS, the selection process closely mirrors the functionality of a roulette wheel, where each individual in the population is assigned a distinct segment on the wheel. The probability of an individual's selection is contingent upon the ratio of their fitness score to the total fitness of the population. In simpler terms, individuals with higher fitness scores are allocated more significant segments on the wheel, augmenting their likelihood of being chosen as parents during the evolutionary process [30]. It is important to note that this mechanism may overlook influential individuals and result in biased selection towards high fitness, not assuring that the best individual will be chosen for the subsequent generation [30].

RWS exhibits an $O(n)^2$ time complexity involving n loops (generations), each iterating through the population. This elucidates the frequency with which the selection process is reiterated for each generation, underscoring the computational overhead associated with this method [26], [29], [31], [32].

2) Tournament Selection: In Tournament Selection (TS), a set of n individuals is randomly chosen from the broader population, instigating a competitive scenario among the selected individuals. The victor, distinguished by the highest fitness, is then incorporated into the mating pool for the subsequent generation population. The mechanism of TS is depicted in Fig. 3, wherein the tournament size (Ts) is designated as three, signifying that three chromosomes engage in competition, with the superior chromosome selected to reproduce the fittest individual [6], [10], [28].

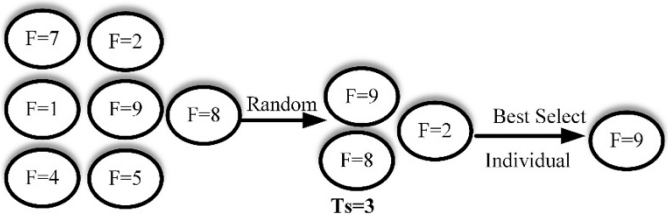

Fig. 3 A population of Individuals

Although the RWS method exhibits greater sophistication compared to TS, its practical application can be excessively time-consuming or virtually unfeasible in scenarios with large populations or parallel population distributions [28], [33]. The time complexity of TS remains *O(nlogn)*.

3) Rank Selection: The population undergoes sorting through Rank Selection (RS), a methodology grounded in fitness worth. In this process, the fitness assignment to each individual is determined by its rank, rendering the actual objective value inconsequential [10], [20], [28]. RS offers advantages by addressing scaling issues inherent in proportional fitness assignment, providing a robust mechanism for controlling selective pressure, and ensuring uniform scaling across the population. However, a notable disadvantage is the additional overhead introduced by the sorting process [15], [34]. Whether employing a Linear Rank or Exponential Rank mapping function, the fundamental concept of rank-based selection remains consistent.

RS comprises two distinct processes. The initial step involves sorting individuals within the population, with the time complexity, assuming the use of an optimal sorting technique, being *O(nlogn)*. The second step is RS, characterized by an *O(nlogn)* time complexity.

4) Stochastic Universal Selection: The SUS proposed by Baker in 1987 [35], is a variant of proportionate selection operator akin to RWS, with a focus on alleviating certain drawbacks associated with RWS, particularly in scenarios involving the selection of multiple parents. SUS maintains selection based on the fitness values of individuals within the population. In contrast to assigning each individual a segment on a conventional roulette wheel, SUS employs a single, elongated roulette wheel, with each segment precisely equal to the corresponding fitness, mirroring the method of roulette-wheel selection. Subsequently, equally spaced pointers, numbering the same as the individuals to be selected, are positioned along the wheel.

Fig. 4 illustrates the procedure of selecting individuals through the SUS method, where the variable N pointers denote the number of individuals to be chosen. The initial position of the pointer is determined by a randomly generated number within the range *[0,1/N Pointer]*, and the intervals between the pointers are set at *1/N Pointer* [25].

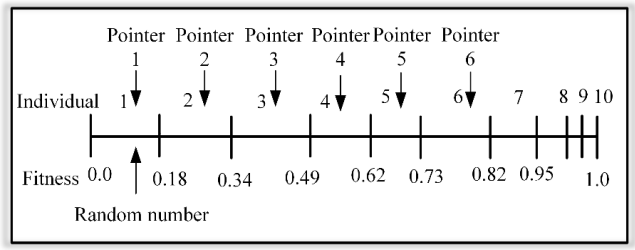

Fig. 4 Stochastic Universal Selection

The primary improvement of SUS over traditional RWS lies in its ability to ensure a more uniform and consistent selection of individuals, mitigating the risk of disproportionately favoring high-fitness individuals. Multiple evenly spaced pointers enhance the probability of selecting individuals across a broader fitness spectrum [6]. Due to the selection of one element through the population after the summation of fitness values, the time complexity of SUS is generally linear *O(n)*, rendering it an efficient and widely adopted method in GA. SUS proves especially beneficial when selecting multiple parents simultaneously, fostering diversity in the mating pool and contributing to the overall exploration of the solution space [31].

5) Proposed Selection Operator: Various selection methods aim to choose a subset of individuals based on their fitness values rather than considering the entire population [26]. The Selection stage can be time-consuming, particularly with a large population size [4], [36], [37], [38]. RS consistently demonstrates superior solution quality [25], [28], [37]. In RS, the population is sorted, with the best fitness corresponding to position 0 and the worst fitness associated with position N. Selecting this position number results in a high fitness proportion. RS involves two steps: initially sorting the population size list based on fitness values and then ranking the individuals. The dual-process nature of RS,

with its two sorting operations, contributes to increased computational complexity. Previous research [33], [35] has identified sorting techniques with a time complexity of O(nlogn). SUS exhibits lower time complexity than RS as it chooses parents randomly, eliminating the need for a sorting process. However, this randomness introduces the risk of selecting lower-quality parents [28].

The primary objective of this study is to enhance the performance of the GA by introducing a novel parameter in SUS, aimed explicitly at improving solution quality for the TSP within a reasonable timeframe. The conceptual foundation of this innovative selection operator revolves around identifying the optimal fitness value among pre-established solutions. A designated proportion (P) is extracted from the optimal fitness in this phase. Subsequently, this proportion (*P*) is added to the optimal fitness, establishing a new fitness value. Then, rather than dividing the fitness value of the nominated individual by the total number of organisms (population size), as in SUS, the fitness value of the selected individual is divided by the new fitness value. For example, consider Table 1, which encompasses all pre-established solutions, and assume the optimal fitness among them is 100. Suppose we extract 50% from this optimal fitness value, denoting P as 50. In that case, we subsequently add this proportion (*P*) to the optimal fitness value, establishing a new fitness value equal to 150.

TABLE I
THE PROBABILITY OF THE SELECTION (PS) USING THE ORIGINAL SUS AND NEW SELECTION OPERATOR

No	Individual Fitness Value	Original Sus	New Selection Operator
1	200	200/830=0.240	200/150=1.33
2	140	140/830=0.168	140/150=0.93
3	120	120/830=0.144	120/150=0.80
4	100	100/830=0.120	100/150=0.66
5	270	270/830=0.325	270/150=1.80
SUM	830	0.990	5.22

The distinction between SUS and the proposed selection operator lies in the methodology of individual selection. In the original SUS, the probability for individual i to be selected for mating in a population of size u is computed as in Equation 2 [20], [37].

$$ps(i) = \frac{f_i}{\sum_{j=1}^{i=u} f_i} \quad (2)$$

where j=1,2...u, f_i represents the fitness value of an individual i, typically selected through a random process, while f_j is the total fitness of the population.

While the probability for individual i to be selected for mating in a population of size u is computed in the proposed selection operator is the fitness value of an individual as by Equation 3:

$$ps(i) = \frac{f(i)_{rand}}{f_{best}+p} \quad (3)$$

where $f(i)_{rand}$ represents the fitness, typically selected randomly, $f_{best}+p$ is the best fitness in the present population, and p is the proportion added to the best fitness. Normally, *p* is in the order of 10% – 50%; however, its value is contingent upon the representation of individual solutions and the number of individual solutions selected for generating the next generation

The objective of choosing the new fitness is to move towards optimal solutions and effectively narrow the search space, as shown in Equation 4. This novel selection operator randomly selects any individual with a value less than 1, as shown in Fig. 5. Although similar to RS, it operates without the need for sorting. In contrast to the original SUS that selects any individual, the outcome may vary based on the initial selection. Additionally, time complexity calculations treat these as distinct steps. After finding the best fitness within the population, the time complexity is *O(n)*. Similarly, the time complexity of the new selection operator for selection is *O(n)*.

$$\text{The proposed selection} = \begin{cases} best\ fitness\ value, & if\ 0 < ps < 1 \\ worst\ fitness\ value, & if\ ps \geq 1 \end{cases} \quad (4)$$

This implies that the proposed approach will randomly choose fitness values from any number within the range *[0,1]*. Subsequently, all the remaining individuals are considered inferior, characterized by fitness values equal to 1 or greater.

Algorithm 1: "C" CODE FRAGMENT FOR PROPOSED SELECTION

Input: f_{best} is the minimum fitness within the population initialization. p is $f_{best}+(f_{best} * 50\%)$;
Output: To randomly select fitness values uniformly distributed within the range [0, 1].

```
organism ← random();
ptr ← (organism%Num Of Organisms); //The initial pointer position is determined by a randomly
while i = ptr and i < Num Of Organisms do
    if (fitnesses[ptr]/p< 1) then
        SelectInd[i];
    ++i;
```

Fig. 5 The proposed selection operator

E. Genetic Operators

Genetic operators consist of the Crossover Operation and Mutation Operation. These operations promote diversity and drive the population towards novelty, thereby augmenting the quality of solutions in the new generation through the iterative application of genetic operators [20].

1) Crossover operation: Crossover Operations (CX), a pivotal operator in GA, generates a new offspring by combining genetic material from more than one parent. Order Crossover (OX), Single Point Crossover, Two Point Crossover, Cycle Crossover, and Partially Mapped Crossover (PMX) are all instances of CX [20]. In this study, the One-Point Crossover method, traditionally employed in GA, was utilized. The initial step involves selecting two parents for each child. Subsequently, the GA randomly selects a crossover point and initiates the transfer of parental genes to the offspring. The One-Point Crossover Operator is illustrated in Fig. 6, where genes to the left of the crossover point are inherited from parent one, while genes to the right are inherited from parent two [20], [30].

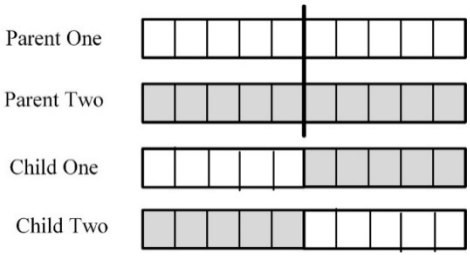

Fig. 6 One-Point of Crossover Operator

2) Mutation operation: The Mutation Operator plays a pivotal role in maintaining population diversity and introducing variations to individual solutions. A spectrum of mutation operations, such as Swap Mutation, Insert Mutation, and Scramble Mutation [20], [39], can be employed, each directed at a single solution (genotype). Specifically, Swap Mutation involves the random selection of a location within the solution, introducing changes and generating new values that are not present in the original sequence, potentially leading to improved results. In the context of this study, the Swap Mutation Operator is implemented to counter premature convergence and enhance population diversity [40]. In Swap Mutation, two cities within the tour are randomly selected, and a subsequent exchange of these two chosen cities takes place (see Fig. 7) [20], [30].

3)

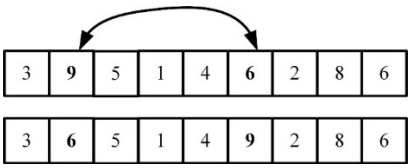

Fig. 7 Swap Mutation Operator Sampling

III. Results and Discussion

This study conducted computational experiments using five distinct cases from the Traveling Salesman Problem Library (TSPLIB) [12]. The primary aim was to assess the performance of the EGA in comparison to the traditional GA. Additionally, symmetric TSP was employed in computations and analyses involving a destination matrix. This ensures that the travel fee from city A to city B is identical to that from B to A, facilitating straightforward distance verification. The selection of parameters is a critical consideration as it directly impacts both the execution time and the quality of the solution. Table 2 delineates these parameters, including the dataset extracted from the TSPLIB library.

The algorithm was implemented in C programming language and executed on a server featuring an Intel Xeon Processor E5-2620 V2 @ 2.10GHz, 8GB RAM, and the Linux operating system. The dataset and parameters are essential to the testing of the GA. Modifying these parameters directly impacts solution quality and the algorithm's runtime. As a stochastic algorithm, GA derives its complexity from factors such as the number of generations, population size, chromosome representation, and fitness function calculation. The sequential time execution of GA is determined using the high-resolution function *gettimeofday()*.

TABLE III
Parameters of the EGA and Traditional GA

Parameters	Value
TSP Name	gr120, brazil58, gr24, si175 and pa561
Population Size	1024, 2048, 4096
Type of Parent Selection	SUS and proposed selection operator
The number of iterations (as ending criterion)	1000
Type of Crossover	One-Point
Type of Mutation	Swap

The experiment conducted ten trials on each TSP dataset to assess solution quality and time complexity. Subsequently, the average tour length and execution time were computed, and results obtained using the SUS method (the original algorithm) and the proposed selection operator were compared. The trade-off between average tour length and average execution time is detailed in Tables 3 and illustrated in Fig. 8 and Fig. 9.

TABLE IIIII
Average execution time (in seconds) and average tour length using the proposed selection operator and SUS for selection

TSP Name	Population Size	Traditional GA		EGA	
		Tour length	Execution time(s)	Tour length	Execution time(s)
gr120	1024	1323	0.9	1258	0.85
	2048	1314	2.2	1313	2.1
	4096	1361	11.9	1320	10.6
brazil58	1024	99887	4.0	33695	3.7
	2048	103250	18.3	70800	13.4
	4096	105050	44.1	84443	36.7
gr24	1024	49281	13.8	27919	11.2
	2048	46898	45.4	29275	43.6
	4096	48955	204.7	29959	161.6
si175	1024	45068	16.9	33652	15.0
	2048	46450	69.1	33590	66.2
	4096	46968	168.4	34803	156.7
pa561	1024	25814	636.6	10637	619.7
	2048	25405	1278.8	16905	1232.0
	4096	25359	2626.8	18191	2539.5

Table 3 presents the results of experiments conducted on the traditional GA and EGA. The primary objectives of these experiments were to determine the average tour length and average execution times using both the traditional GA and the EGA. Furthermore, the performance evaluation encompassed five TSP samples (datasets) with 24, 58, 120, 175, and 561 cities (indicating the length of the solution) and population sizes of 1024, 2048, and 4096 for each dataset. The execution involved 10 trials on each TSP dataset.

Figure 8 depicts the representations of the five TSP samples, where cities are symbolized by the columns, and the solutions are represented by the lengths of the columns. Various population sizes are considered for both the traditional GA and the EGA. Meanwhile, Fig. 9 illustrates the representations of average execution times for both the traditional GA and the EGA at population sizes of 1024, 2048, and 4096.

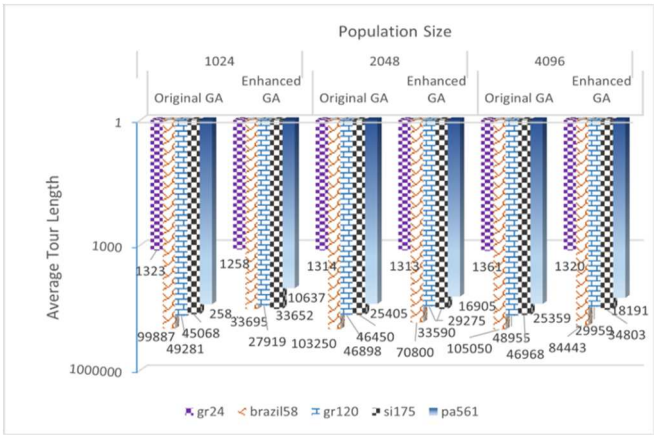

Fig. 8 The results of the tour length of both the traditional GA and EGA at population sizes of 1024, 2048, and 4096

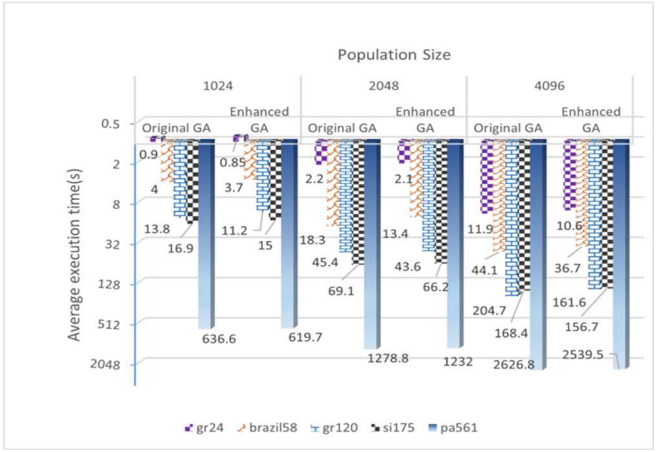

Fig. 9 The results of Average execution time(s) of both the traditional GA and EGA at population sizes of 1024, 2048, and 40962048, and 4096

The statistical analysis also evaluates the significant difference between the traditional GA and the EGA. The t-test, a commonly utilized statistical method in research [41], is a fundamental tool for comparing means and discerning statistically significant differences between two groups. This study used the t-test to assess the significance of differences between the traditional GA and the EGA, focusing on key metrics such as execution time and tour length. The significance level, denoted as α, was set at 0.05, a standard threshold for hypothesis testing. A significant difference was concluded between the traditional GA and the EGA if the p-value was more remarkable than α. Conversely, if the p-value was less than α, no discernible difference existed between the two algorithms. From the t-test, it was found that all the p-values for both execution time and tour length were greater than α. This rigorous statistical analysis provided a robust foundation for evaluating the efficacy of the enhancements incorporated into the EGA.

The results of these experiments underscore the efficacy of both algorithms in addressing the challenges posed by the TSP. Notably, both traditional GA and the EGA delivered satisfactory results, especially when confronted with smaller population sizes and a limited number of cities. Nevertheless, as the complication of the problem increased with a larger number of cities, the EGA consistently outperformed the traditional GA. This persistent superiority of the EGA over the traditional GA indicates substantial enhancements in result quality, especially while maintaining the running time, as depicted in Table 3, Fig. 8, and Fig. 9.

IV. Conclusion

Both selection and genetic operators play a pivotal role in determining the functioning of a GA, as these are two fundamental factors essential to its functionality. The first factor, selection, is responsible for choosing the set of chromosomes for mating. In contrast, the additional factor involves genetic operators (crossover and mutation), which are employed to generate new individuals and introduce random changes. This article introduces a novel parameter in SUS, aiming to effectively narrow the search space and optimize the genetic operators for selecting promising candidate solutions (parent selection). Consequently, the proposed approach selects individuals based on their fitness scores, addressing population sorting and individual ranking issues while mitigating computational complexity. To evaluate the performance of the proposed operator, experiments were handled with both the traditional GA and the EGA on a well-known TSPLIB benchmark dataset, encompassing various problem sizes. The EGA consistently outperforms the sequential traditional GA in terms of solution quality. These findings suggest potential avenues for future research, exploring the algorithm's behavior with larger datasets and enhanced genetic operators (crossover and mutation). Additionally, the presented study could be expanded to enhance solution quality by investigating the association between the proportion included to the best fitness and population size to achieve improved consistency. It is hoped that the proposed method can be utilized in different areas such as image recognition [42], [43], cyber security [44], [45], [46], [47], healthcare [48], [49] and machine learning [50], [51].

Acknowledgment

Research Grant the Ministry of Higher Education Malaysia provided funding for this study through the Fundamental Research Grant (FRGS) with FRGS/1/2022/ICT02/UMP/02/2 (RDU220134).

References

[1] R. W. Dewantoro, P. Sihombing, and others, "The combination of ant colony optimiza-tion (ACO) and tabu search (TS) algorithm to solve the traveling salesman problem (TSP)," in 2019 3rd International Conference on Electrical, Telecommunication and Computer Engineering (ELTICOM), 2019, pp. 160–164.

[2] Y. Shi and Y. Zhang, "The neural network methods for solving Traveling Salesman Prob-lem," Procedia Comput Sci, vol. 199, pp. 681–686, 2022, doi: 10.1016/j.procs.2022.01.084.

[3] S. Prayudani, A. Hizriadi, E. B. Nababan, and S. Suwilo, "Analysis effect of tournament se-lection on genetic algorithm performance in traveling salesman problem (TSP)," in Journal of Physics: Conference Series, 2020, p. 12131.

[4] H. R. Er and N. Erdogan, "Parallel Genetic Algorithm to Solve Traveling Salesman Prob-lem on MapReduce Framework using Hadoop Cluster," Jscse, vol. 3, no. 3, pp. 380–386, 2013, doi:10.7321/jscse.v3.n3.57.

[5] A. R. S. K. Hegde, "A Novel Method to Solve Travelling Salesman Problem Using Sequen-tial Constructive Crossover Using Map/Reduce Framework," International Journal of Science and Research (IJSR), vol. 4, no. 5, pp. 1362–1367, 2015.

[6] S. Katoch, S. S. Chauhan, and V. Kumar, "A review on genetic algorithm: past, present, and future," Multimed Tools Appl, vol. 80, no. 5, pp. 8091–8126, Feb. 2021, doi: 10.1007/s11042-020-10139-6.

[7] Z. Drezner and T. D. Drezner, "Biologically Inspired Parent Selection in Genetic Algo-rithms," Ann Oper Res, vol. 287, no. 1, pp. 161–183, Apr. 2020, doi: 10.1007/s10479-019-03343-7.

[8] C. Nilsson, "Heuristics for the traveling salesman problem," 2003. doi:10.1016/S0305-0548(98)00085-9.

[9] G. Steven, "Evolutionary algorithms for single and multicriteria design optimization. A. Osyczka. Springer Verlag, Berlin, 2002, ISBN 3-7908-1418-01," Structural and Multidisciplinary Optimization, vol. 24, no. 1, pp. 88–89, Aug. 2002, doi: 10.1007/s00158-002-0218-y.

[10] A. Hussain and Y. S. Muhammad, "Trade-off between exploration and exploitation with genetic algorithm using a novel selection operator," Complex & intelligent systems, vol. 6, no. 1, pp. 1–14, 2020.

[11] Z. H. Ahmed, "Adaptive Sequential Constructive Crossover Operator in a Genetic Algo-rithm for Solving the Traveling Salesman Problem," International Journal of Advanced Computer Science and Applications, vol. 11, no. 2, 2020, doi: 10.14569/ijacsa.2020.0110275.

[12] G. Reinhelt, "TSPLIB: a library of sample instances for the TSP (and related problems) from various sources and of various types," 2014.

[13] C.-W. Tsai, S.-P. Tseng, M.-C. Chiang, C.-S. Yang, and T.-P. Hong, "A High-Performance Genetic Algorithm: Using Traveling Salesman

Problem as a Case," The Scientific World Jour-nal, vol. 2014, pp. 1–14, 2014, doi: 10.1155/2014/178621.
[14] Z. H. Ahmed, "Genetic Algorithm for the Traveling Salesman Problem using Sequential Constructive Crossover Operator," Int J Biom Bioinformatics, vol. 3, no. 6, pp. 96–105, 2010.
[15] A. Aranganayaki, "Reduce Total Distance and Time Using Genetic Algorithm in Traveling Salesman Problem," International Journal of Computer Science & Engineering Technology, vol. 5, no. 2229–3345, p. 4, 2014.
[16] K. Rani and V. Kumar, "Solving travelling Salesman problem using genetic algorithm based on heuristic crossover and mutation operator," Int J Res Eng Technol, vol. 2, no. 2, pp. 27–34, 2014.
[17] A. Phu-ang, "The new technique based on the galaxy based search algorithm for solving the symmetric travelling salesman problem," in 2018 International ECTI Northern Section Con-ference on Electrical, Electronics, Computer and Telecommunications Engineering (ECTI-NCON), 2018, pp. 131–134. doi: 10.1109/ecti-ncon.2018.8378296.
[18] Y. Gao and J. Ye, "An Improved Genetic Algorithm Based on Normal Distribution for Solving the Traveling Salesman Problem," in 2018 International Conference on Virtual Reality and Intelligent Systems (ICVRIS), 2018, pp. 360–362. doi: 10.1109/icvris.2018.00094.
[19] M. L. Islam, D. Pandhare, A. Makhthedar, and N. Shaikh, "A Heuristic Approach for Opti-mizing Travel Planning Using Genetics Algorithm," International Journal of Research in Engi-neering and Technology eISSN: 2319-1163, pISSN: 2321, vol. 7308, no. 01, 2014.
[20] A. E. Eiben and J. E. Smith, Introduction to Evolutionary Computing. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015.
[21] X. Liu, M. Zhang, Z. Bai, L. Wang, W. Du, and Y. Wang, "Function Call Flow based Fitness Function Design in Evolutionary Testing," in 14th Asia-Pacific Software Engineering Conference (APSEC'07), 2007, pp. 57–64. doi: 10.1109/aspec.2007.13.
[22] A. Rao and S. K. Hegde, "Literature Survey On Travelling Salesman Problem Using Genet-ic Algorithms," 2015.
[23] S. S. Juneja, P. Saraswat, K. Singh, J. Sharma, R. Majumdar, and S. Chowdhary, "Travelling Salesman Problem Optimization Using Genetic Algorithm," in 2019 Amity International Con-ference on Artificial Intelligence (AICAI), 2019, pp. 264–268. doi:10.1109/aicai.2019.8701246.
[24] M. S. H. Kalathingal, S. Basak, and J. Mitra, "Artificial neural network modeling and genetic algorithm optimization of process parameters in fluidized bed drying of green tea leaves," J Food Process Eng, vol. 43, no. 1, p. e13128, Jan. 2020, doi: 10.1111/jfpe.13128.
[25] I. Jannoud, Y. Jaradat, M. Z. Masoud, A. Manasrah, and M. Alia, "The Role of Genetic Al-gorithm Selection Operators in Extending WSN Stability Period: A Comparative Study," Electronics (Basel), vol. 11, no. 1, p. 28, Dec. 2021, doi: 10.3390/electronics11010028.
[26] S. N. Sivanandam and S. N. Deepa, Introduction to Genetic Algorithms. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008. doi:10.1007/978-3-540-73190-0.
[27] J. Too and A. R. Abdullah, "A new and fast rival genetic algorithm for feature selection," J Supercomput, vol. 77, no. 3, pp. 2844–2874, Mar. 2021, doi: 10.1007/s11227-020-03378-9.
[28] M. Noraini and J. Geraghty, "Genetic algorithm performance with different selection strategies in solving TSP," in World Congress on Engineering, 2011, pp. 4–9.
[29] M. Gen and R. Cheng, Genetic Algorithms and Engineering Optimization, vol. 7. Wiley, 1999. doi: 10.1002/9780470172261.
[30] S. Mirjalili, "Genetic Algorithm," in Evolutionary Algorithms and Neural Networks: Theory and Applications, Springer, 2019, pp. 43–55. doi: 10.1007/978-3-319-93025-1_4.
[31] D. E. Goldberg and K. Deb, "A Comparative Analysis of Selection Schemes Used in Ge-netic Algorithms," in Foundations of genetic algorithms, vol. 1, 1991, pp. 69–93.
[32] K. Asghari, M. Masdari, F. S. Gharehchopogh, and R. Saneifard, "Multi-swarm and chaotic whale-particle swarm optimization algorithm with a selection method based on roulette wheel," Expert Syst, vol. 38, no. 8, p. e12779, Dec. 2021, doi: 10.1111/exsy.12779.
[33] T. Blickle and L. Thiele, "A Comparison of Selection Schemes used in Genetic Algo-rithms," Evol Comput, vol. 2, no. 11, pp. 311–347, 1995, doi: 10.1162/evco.1996.4.4.361.
[34] M. Mitchell, "Genetic algorithms: An overview," Complexity, vol. 1, no. 1, pp. 31–39, Sep. 1995, doi: 10.1002/cplx.6130010108.
[35] J. E. Baker, "Reducing bias and inefficiency in the selection algorithm," in Proceedings of the second international conference on genetic algorithms, 1987, pp. 14–21.
[36] T. Harada and E. Alba, "Parallel Genetic Algorithms," ACM Comput Surv, vol. 53, no. 4, pp. 1–39, Jul. 2021, doi: 10.1145/3400031.
[37] H. M. Pandey, "Performance Evaluation of Selection Methods of Genetic Algorithm and Network Security Concerns," Procedia Comput Sci, vol. 78, no. December 2015, pp. 13–18, 2016, doi:10.1016/j.procs.2016.02.004.
[38] M. Abbasi, M. Rafiee, M. R. Khosravi, A. Jolfaei, V. G. Menon, and J. M. Koushyar, "An efficient parallel genetic algorithm solution for vehicle routing problem in cloud imple-mentation of the intelligent transportation systems," Journal of Cloud Computing, vol. 9, no. 1, p. 6, Dec. 2020, doi: 10.1186/s13677-020-0157-4.
[39] Z. Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs. Berlin, Heidelberg: Springer Berlin Heidelberg, 1996. doi:10.1007/978-3-662-03315-9.
[40] A. Plichta, T. Gaciarz, B. Baranowski, and S. Szominski, "Implementation Of The Genetic Algorithm By Means Of CUDA Technology Involved In Travelling Salesman Problem," in ECMS 2014 Proceedings edited by: Flaminio Squazzoni, Fabio Baronio, Claudia Archetti, Marco Castellani, 2014, pp. 475–479. doi:10.7148/2014-0475.
[41] S. García, D. Molina, M. Lozano, and F. Herrera, "A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the CEC'2005 Special Session on Real Parameter Optimization," Journal of Heuristics, vol. 15, no. 6, pp. 617–644, Dec. 2009, doi: 10.1007/s10732-008-9080-4.
[42] K. Moorthy, K. Mohd Daud, S. R. Arokiasamy, and M. R. I. Tomal, "Hybrid Biometric Au-thentication for Automatic Teller Machine," International Journal of Software Engineering and Computer Systems, vol. 10, no. 1, pp. 32–39, Sep. 2024, doi:10.15282/ijsecs.10.1.2024.3.0121.
[43] Y. Zhong, K. M. Daud, A. N. B. M. Nor, R. A. Ikuesan, and K. Moorthy, "Offline Handwrit-ten Chinese Character Using Convolutional Neural Network: State-of-the-Art Methods," Journal of Advanced Computational Intelligence and Intelligent Informatics, vol. 27, no. 4, pp. 567–575, 2023, doi: 10.20965/jaciii.2023.p0567.
[44] A. Nuhu, A. F. Mat Raffei, M. F. Ab Razak, and Abubakar Ahmad, "Distributed Denial of Service Attack Detection in IoT Networks using Deep Learning and Feature Fusion: A Re-view," Mesopotamian Journal of CyberSecurity, vol. 4, no. 1, pp. 47–70, Apr. 2024, doi:10.58496/mjcs/2024/004.
[45] M. I. Jaya and M. F. Ab. Razak, "Dynamic Ransomware Detection for Windows Platform Using Machine Learning Classifiers," JOIV : International Journal on Informatics Visualization, vol. 6, no. 2–2, p. 469, Aug. 2022, doi: 10.30630/joiv.6.2-2.1093.
[46] N. S. Nordin and M. A. Ismail, "A hybridization of butterfly optimization algorithm and harmony search for fuzzy modelling in phishing attack detection," Neural Comput Appl, vol. 35, no. 7, pp. 5501–5512, Mar. 2023, doi: 10.1007/S00521-022-07957-0/tables/6.
[47] E. H. Tusher, M. A. Ismail, M. A. Rahman, A. H. Alenezi, and M. Uddin, "Email Spam: A Comprehensive Review of Optimize Detection Methods, Challenges, and Open Research Problems," IEEE Access, vol. 12, pp. 143627–143657, 2024, doi:10.1109/access.2024.3467996.
[48] M. A. I. Rohismadi, A. F. M. Raffei, N. S. A. Zulkifli, M. H. Ithnin, and S. F. Othman, "An Automated Strabismus Classification Using Machine Learning Algorithm for Binocular Vi-sion Management System," in 2023 IEEE 8th International Conference On Software Engineering and Computer Systems (ICSECS), 2023, pp. 487–492. doi: 10.1109/icsecs58457.2023.10256291.
[49] A. F. Z. Abidin et al., "Adaboost-multilayer perceptron to predict the student's perfor-mance in software engineering," Bulletin of Electrical Engineering and Informatics, vol. 8, no. 4, pp. 1556–1562, 2019, doi:10.11591/eei.v8i4.1432.
[50] A. Feizollah, N. B. Anuar, R. Mehdi, A. Firdaus, and A. Sulaiman, "Understanding COVID-19 Halal Vaccination Discourse on Facebook and Twitter Using Aspect-Based Sentiment Analysis and Text Emotion Analysis," Int J Environ Res Public Health, vol. 19, no. 10, 2022, doi: 10.3390/ijerph19106269.
[51] N. F. Idris, M. A. Ismail, M. I. M. Jaya, A. O. Ibrahim, A. W. Abulfaraj, and F. Binzagr, "Stacking with Recursive Feature Elimination-Isolation Forest for classification of diabetes mellitus," PLOS ONE, vol. 19, no. 5, p. e0302595, May 2024, doi:10.1371/journal.pone.0302595.