applied sciences

MDPI

Article

An Efficiency Boost for Genetic Algorithms: Initializing the GA with the Iterative Approximate Method for Optimizing the Traveling Salesman Problem—Experimental Insights

Esra'a Alkafaween [1], Ahmad Hassanat [1,*], Ehab Essa [2] and Samir Elmougy [2]

[1] Computer Science Department, Mutah University, Mutah 61710, Jordan; esra_ok@mutah.edu.jo
[2] Department of Computer Science, Mansoura University, Mansoura 35516, Egypt; ehab_essa@mans.edu.eg (E.E.); mougy@mans.edu.eg (S.E.)
* Correspondence: hasanat@mutah.edu.jo

Abstract: The genetic algorithm (GA) is a well-known metaheuristic approach for dealing with complex problems with a wide search space. In genetic algorithms (GAs), the quality of individuals in the initial population is important in determining the final optimal solution. The classic GA using the random population seeding technique is effective and straightforward, but the generated population may contain individuals with low fitness, delaying convergence to the best solution. On the other side, heuristic population seeding strategies provide the advantages of producing individuals with high fitness and encouraging rapid convergence to the optimal solution. Using background information on the problem being solved, researchers have developed several population seeding approaches. In this paper, to enhance the genetic algorithm efficiency, we propose a new method for the initial population seeding based on a greedy approach. The proposed method starts by adding four extreme cities to the route, creating a loop, and then adding each city to the route through a greedy strategy that calculates the cost of adding every city to different locations along the route. This method identifies the best position to place the city as well as the best city to add. Employing local constant permutations improves the resultant route even more. Together with the suggested approach, we examine the GA's effectiveness while employing conventional population seeding methods like nearest-neighbor, regression-based, and random seeding. Utilizing some of the well-known Traveling Salesman Problem (TSP) examples from the TSPLIB, the standard library for TSPs, tests were conducted. In terms of the error rate, average convergence, and time, the experimental results demonstrate that the GA that employs the suggested population seeding technique performs better than other GAs that use conventional population seeding strategies.

Keywords: genetic algorithms; initial population; TSP; IAM-TSP+

Citation: Alkafaween, E.; Hassanat, A.; Essa, E.; Elmougy, S. An Efficiency Boost for Genetic Algorithms: Initializing the GA with the Iterative Approximate Method for Optimizing the Traveling Salesman Problem—Experimental Insights. *Appl. Sci.* **2024**, *14*, 3151. https://doi.org/10.3390/app14083151

Academic Editor: Richard C. Millham

Received: 4 March 2024
Revised: 27 March 2024
Accepted: 29 March 2024
Published: 9 April 2024

1. Introduction

Genetic algorithms (GAs) are stochastic optimization methods based on principles of biological organisms: survival of the fittest and natural selection [1–3]. Numerous combinatorial optimization issues in business, engineering, and social sciences have been successfully solved with GAs [4–7]. A GA has two basic characteristics: it is a randomized algorithm in which choosing and reproducing occur at random. The GA always works with a set of solutions; this gives the approach the advantage of variation, along with durability [8,9]. The GA aims to find the best solution within a large search space that includes all feasible solutions [10]. A chromosome (or individual) is a representation of a potential and feasible solution to a problem [11]. A population is a group of individuals that can be managed by the GA at a certain point in time. An individual's fitness is the value associated with the corresponding individual. The fitness function, which is problem-specific, refers to evaluating an individual's quality to determine the most effective solution to the problem at hand. Finding a superior solution or exceeding the maximum generation

limit can be used as a termination criterion for the GA [12]. The GA begins with seeding the population in the first step, followed by the selection process for individuals, and the third step consists of the crossover and mutation processes [13]. The initial step is performed only once, while the remaining steps continue whenever the final-step criterion is satisfied [14]. Figure 1 illustrates the construction of a common genetic algorithm [15].

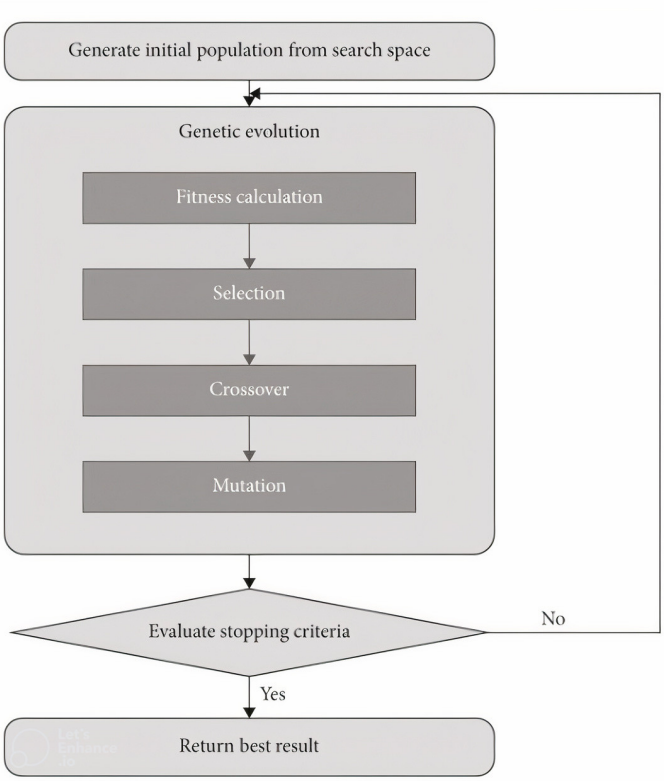

Figure 1. The standard procedure of genetic algorithms [15].

1.1. Initial Population

The first phase of every GA is the population seeding phase. As the input for the GA, it produces a population of possible solutions or individuals at random or through a heuristic initialization technique [16]. Because the quality of individual solutions created in the initial population stage is significant in determining the quality of the ultimate optimal solution, population initialization is an important phase in the GA [17]. Random initialization is the most frequent approach used to generate the initial population in a GA [13,18,19]. The random approach has a low-fitness solution, which reduces the chances of discovering an optimal or near-optimal solution; it also takes a considerable amount of search time if information is lacking [20,21]. Thus, it is clear that an improved population seeding approach is needed for the GA, and numerous research studies have been undertaken to support this argument. In a large search space, on the other hand, if prior heuristic knowledge about the best solution is provided, it may simply build the initial population and identify high-quality solution regions. The use of heuristic approaches to generate initial population seeding results in a high-quality population, allowing GAs to identify better solutions quicker. However, it is possible that it will end up with a narrow search space and will never be able to find the global optimal solution [22]. Since the inception of GA ideas, a variety of initialization strategies have been introduced, such as the following:

- Nearest neighbor (NN) [23];
- Selective initialization (SI) [24];
- Gene bank (GB) [25];
- Equi-begin Vari-diversity (EV) [8];

- Sorted population (SP) [26];
- K-means initial population [27];
- Knowledge-based initialization [28];
- Ordered Distance Vector population seeding [27].

1.2. TSP

The TSP is a combinatorial optimization problem [29] that is simple to explain but difficult to solve, and it is categorized as one of the problems that cannot be solved in polynomial time; in other words, it is an NP-hard issue [30].
The TSP's purpose is to determine a complete, cheap tour in which a salesman must visit each of the n provided cities only once. The TSP's search space consists of permutations of n cities. Each combination of n cities generates a solution, and given n cities, the size of the overall TSP search space is $s((n-1)!/2)$ [8,31].

Due to its importance and application in multiple areas of daily life, the common NP-hard problem known as the TSP has been comprehensively and widely explored in many fields, especially across the domains of computer science and operations studies; however, it remains an open challenge, and the TSP is widely regarded as a standard testbed of multiple combinatorial optimization techniques [27,32].

However, one can question whether the TSP issue is significant enough to warrant all of the attention it has received. Much attention has been paid to the problem because it is a reasonably straightforward problem, and it is so easy to describe but so hard to solve. The TSP is important because it is a representation of a wider category of problems defined as combinatorial optimization problems. The TSP belongs to the group of problems identified as NP-complete problems. In addition, if a polynomial-time-efficient method can be developed for the TSP, then efficient algorithms might be developed for every other NP-complete problem. There are, however, significant examples of actual difficulties that may be expressed as TSPs, and many additional problems are extensions of this problem [33], such as the drilling of printed circuit boards, the overhaul of gas turbine engines, the design of global navigation satellite system surveying networks, wallpaper cutting, computer wiring, X-ray crystallography, vehicle routing, etc. [29,34].

Many techniques for solving the TSP have been proposed, including exact and approximation algorithms [35]. Exact approaches are not generally appropriate for large-scale TSPs. On the contrary, approximation algorithms, particularly many bioinspired algorithms, may produce acceptable solutions to many NP-hard problems in a (reasonably) short running time [10,29,36], such as genetic algorithms, ant colony optimization (ACO), particle swarm optimization (PSO), and artificial neural networks (ANNs) [37,38].

Brady [39] presented the first GA for the TSP in 1985. Over time, the GA implementations evolved from being general to being more specialized to the TSP. Many population seeding, encoding, mutation, and crossover methods were developed throughout the years, further specializing GAs for the TSP [40].

1.3. Proposed Methodology and Contributions

In this paper, a new approach to initializing the population of genetic algorithms (GAs) is presented with the goal of improving the quality of the solutions. The Iterative Approximation Method (IAM-TSP) is the main component of our solution that solves the Traveling Salesman Problem (TSP). This approach, which we described in full in our earlier work [41], yields an approximate polynomial solution for the TSP and acts as the basis for initializing the GA.

The IAM-TSP approach starts with the identification of four extreme cities along the route. Each city is then successively incorporated into the route by comparing and calculating the costs at different insertion sites. By using this approach, the overall path cost is reduced by placing each city in the best possible position. We next apply local constant permutations to the IAM-TSP outputs to further improve the quality and efficiency of the solution, yielding an improved final path called IAM-TSP+.

This work makes several important and varied contributions, mainly to the field of improved GA performance by means of a sound initial population approach. The particular contributions of this study are as follows:

- Improvement of Quality: By using a new method to initialize the GA population, our proposed method guarantees that the beginning point is closer to the optimal solution, resulting in a significant improvement in the quality of the solution.
- Durability: The proposed method shows resilience in generating high-quality solutions in a variety of TSP instances, highlighting its durability and applicability to various scenarios.
- Computational Efficiency: We accomplish computational efficiency by lowering the temporal complexity involved in locating a near-optimal route in TSPs, which improves the GA's overall performance, allowing it to converge faster. This is made possible by the usage of IAM-TSP+.

We evaluated the proposed algorithms against standard TSP datasets to determine how they would perform in terms of key performance measures. Our promising experimental results show that our new method is effective in terms of error percentage, mean convergence, and time. Among the achievements of this paper are new approaches to initial population creation in the context of solving TSPs as compared to random, NN, and regression strategies.

The rest of this research work is organized into the following structure: Section 2 discusses some relevant background ideas. The proposed approach is explained in Section 3. Section 4 evaluates the experimental outcomes. A brief conclusion is presented in the Section 5.

2. Related Work

Heuristic population seeding strategies offer the advantage of giving high-quality individuals at a preliminary phase and the ability to produce near-optimal solutions in a few generations; however, they have disadvantages regarding the diversity of individuals created, randomness, the opportunity to investigate more search space, and the ability to discover a global optimum [42–45]. On the other hand, the random population seeding method provides benefits such as population variety, the effective investigation of the search space, and the discovery of the best solution. Random seeding has two drawbacks: it takes longer to determine the best solution and produces individuals that have the worst possible sequence. Due to the differences between these two kinds of population seeding techniques, we propose an effective population seeding methodology involving randomness, the diversity of individuals, and a probable sequence.

One well-known alternative method for random population seeding in GAs, particularly for the TSP, is the nearest-neighbor (NN) tour creation heuristic. Using this strategy, individuals in the population seeding are built with the city that is nearest to the current city, and these good individuals may alter the later search in subsequent generations [23]. A greedy GA (GGA) was proposed by Yingzi et al. [25] in which a gene bank (GB) is used for population seeding. By collecting the permutation of 'n' cities according to their distance, the GB is generated. By using the GGA approach, a population of individuals is produced from the GB that is above average in terms of fitness and has a short defining length, and using the TSP, the performance of the GGA is justified.

Yugay et al. [26] proposed an enhanced GA with a sorted starting population based on the idea that better parents produce better children. This strategy generates a huge initial population pool, ranks it based on fitness values, and then excludes a subset of individuals with poor fitness.

A new GA for the TSP was created by Nagata and Soler [46]. The initial population was generated using a local search approach, and the GA was thoroughly evaluated using a large number of small-scale TSP cases.

Shanmugam et al. [9] presented a comprehensive survey of several population seeding techniques used on the TSP. They attempted to evaluate the performance of population seeding strategies such as random, NN, gene bank, selective initialization, and sorted

populations. The performance results of prior population seeding strategies were sorted by the error rate and convergence rate. The results showed that the NN population technique outperformed the other strategies tested. NN develops highly fit individuals, followed by selective initialization and gene bank approaches.

Based on the quality of individuals generated, Paul et al. [32] evaluated the effectiveness of several population seeding approaches, including random, gene bank, nearest-neighbor, selective initialization, and sorted population seeding procedures. Using TSP instances, experiments were conducted. The experimental results indicated the order of performance of various population seeding approaches in terms of the error rate and convergence rate. To successfully solve the TSP, a unique Vari-begin and Vari-diversity population seeding strategy was presented. The Individual Diversity factor, which is a key feature of the Vari-begin and Vari-diversity population seeding approach, refers to the variety in city permutation that each individual creates. As a result, the disadvantage of early convergence is mitigated [32].

For a permutation-coded GA, an effective population seeding method based on Ordered Distance Vectors (ODVs) was suggested using an elitist service transfer strategy. The TSP was selected as the testbed, and tests were carried out on benchmark TSP instances of various sizes. The experimental findings support the assertion that, in terms of the convergence rate, error rate, and convergence time, the suggested strategy performs better than the current initialization methods [8].

Deng et al. [27] developed a strategy known as the k-means initial population strategy, based on the k-means algorithm, to reconstruct the route by linking each cluster. The clusters are first sorted based on how far apart their centers are from one another, allowing the initial population to be made up of random travel paths. In order to connect with their neighbors, clusters disconnect a link at random. In order to improve the efficiency of the GA, Chao et al. [28] developed a knowledge-based initialization approach (KI). KI picks up on the features of evolved populations and makes use of them to direct the evolution of initial populations. With this approach, advanced initial solutions without path crossover can be quickly developed. Different initialization strategies were tested using instances from the TSPLIB. According to the experimental results, the suggested initialization approach improved the GA more than previously published strategies, such as the random initialization, NN, GB, and VV techniques.

To solve the large-scale TSP, an efficient GA with new initial population, crossover, and mutation methods was developed. Through five case studies, ranging from small-scale to large-scale TSPs, the effectiveness of the suggested GA is demonstrated. The basic idea of the suggested approach is to create a chromosome for which two neighboring genes should contain two extremely close cities, sometimes but not always closest to one another [47].

Hassanat et al. [13,22] presented a new regression-based method for GA population seeding in order to address TSPs. The proposed method splits a given TSP instance into smaller sub-problems. This is achieved by using the regression line and its perpendicular line, which enables the cities to be repeatedly clustered into four sub-problems; each city's location establishes which category or cluster it belongs to; the algorithm continues until the sub-problem's size is extremely small.

As shown by the literature reviewed above, there are substantial reasons to identify better initial populations for GAs, such as reducing the GA search time required to discover an optimal solution and reducing the number of generations required to obtain a good solution. Identifying the optimal initial population is crucial for determining the optimal solution, and population diversity is required to prevent the GA's premature convergence.

3. Methods

To boost the genetic algorithm's effectiveness, the IAM-TSP+ [41] method was used as an initial population seeding method for the genetic algorithm. The method has proven effective in finding high-quality solutions to the TSP. The IAM-TSP+ is a greedy algorithm for the TSP, and it is an improvement of the IAM-TSP [41] method. The IAM-TSP begins

by selecting four cities within the input set to act as locations in the east, west, north, and south. The order in which these cities are added to the route list is as follows: east, north, west, and south, with an eastward return. Then, the algorithm recursively adds each of the remainder cities to the route one at a time. Every iteration of the technique determines the cost of adding a city at different places along the route. It thoroughly looks into every possible position for every location, selecting the best position and city for the least cost. Lastly, this method calculates the overall cost of the route and outputs the result. The Euclidean distance between each city along the route is used to compute the cost. The IAM-TSP pseudocode is shown in Algorithm 1.

Algorithm 1 The proposed IAM-TSP algorithm

Require: Cities xy (list of cities with x- and y-coordinates) of size n (number of cities)
Ensure: *Cost* (total cost of the route using Euclidean distance) and *Route* (the best possible sequence of cities)

Create an empty list *Route* to store the order of visited cities.
Obtain the *East*, *North*, *West*, and *South* cities and add them to *Route* .
Initialize a Boolean array *Visited* of size n and set all elements to $false$.
Set the *Visited* status of the cities in *Route* to true (*East*, *North*, *West*, and *South*).
Create an empty list *Open*.
Add all cities in *Citiesxy* to *Open* if their *Visited* status is $false$.
while *Open* is not empty **do**
    Create an empty list *PossibleSolutions* to store the possible solutions (routes).
    for each city i in *Open*
        find the best location to insert i into *Route*
        add the new *Route* to *PossibleSolutions*
    end for
    Find the best solution *Best* in *PossibleSolutions* and keep track of the city ID.
    Update *Route* by *Best*.
    Remove the city ID from *Open* that satisfies *Best*.
end while
Calculate the cost of the *Route* using the Euclidean distance and store it in *Cost*.
return *Cost*, *Route*.

With a predetermined number of local cities (k), an improved version of the IAM-TSP called IAM-TSP+ computes every permutation sequence from the first city in the output route to the kth city, identifying the best solution and updating the route in the process. Following that, the method goes into a loop where it iterates through each city until it finds the next k permutations until $n - k$. The procedure of the suggested IAM-TSP+ is displayed in Algorithm 2.

Algorithm 2 The proposed IAM-TSP+ algorithm

Require: Cities xy of size n and $k = 5$ (local permutations)
Ensure: *Cost*, *Route*

Route = Algorithm1(*Citiesxy*).
for $i = 1$ to $n - k$ **do**
    create an empty list of lists *PossibleSolutions* to store the possible solutions (routes).
    for each j from 1 to k, generate all k local permutations of *Route* from j to $j + k$.
    Update the cost of each solution in *PossibleSolutions*.
    Route = the minimum cost solution in *PossibleSolutions*.
end for
return $Cost(Route)$, *Route*

The IAM-TSP+ route results for a280 and tsp225 from the TSPLIB [48] are shown in Figures 2 and 3. This method provides a single solution; because the population requires many solutions to initiate the GA, we utilized the seed solution from the previous algorithm to generate n solutions using the mutation operator, which mutates the seed solution $n-1$ times, in which n is the size of the population. To boost the diversity of the starting population, we employed the mutation operator rather than the crossover operator in this case. The GA will then follow up to optimize the solutions after the initial population is finished. In this work, we used several types of mutations to achieve a kind of diversity. We used IRGIBNNM mutation [14], flip mutation [49], and slide mutation [50].

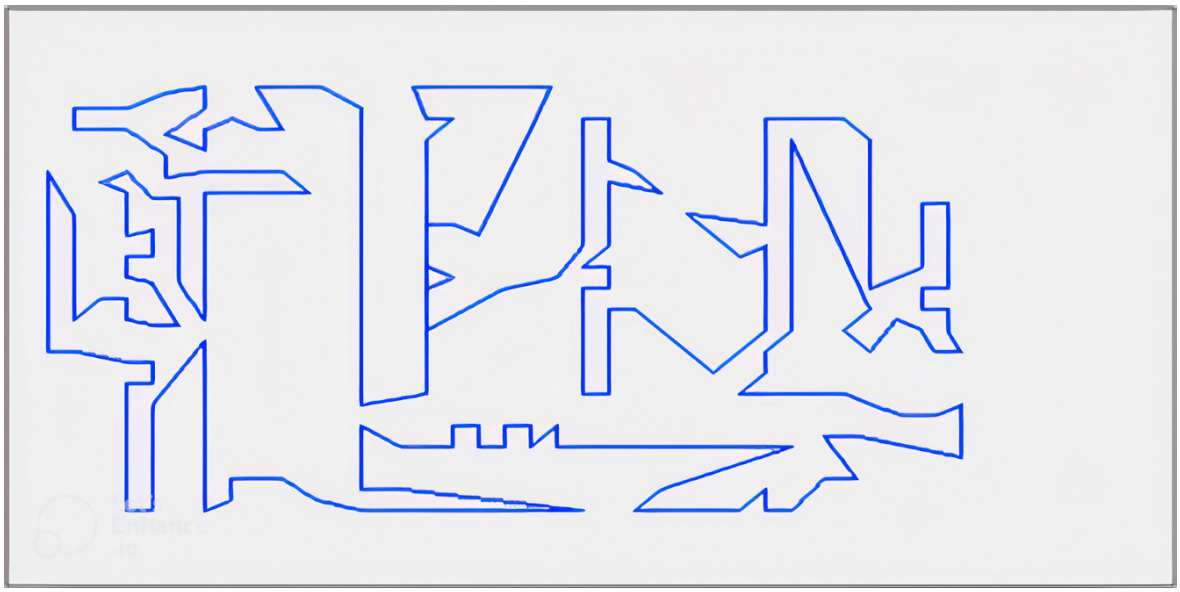

Figure 2. The initial solution for a280.

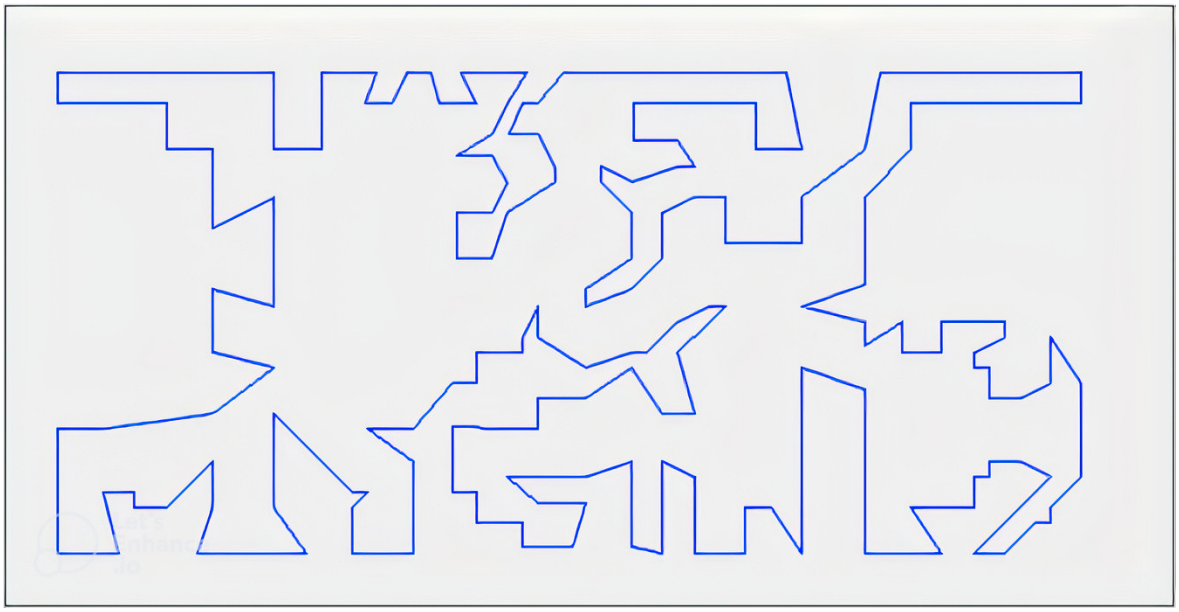

Figure 3. The initial solution for tsp225.

4. Experimental Settings, Results, and Discussion

Investigations were carried out using TSP instances from the TSPLIB [48], and the TSP was chosen for several reasons, including the fact that it can be used to simulate a wide range of real-world problems, and it serves as a typical testbed for new algorithmic

approaches; success on the TSP is frequently seen as evidence of a method's applicability or effectiveness. In addition, it is simple to understand, preventing excessive complexity from obscuring the behavior of the algorithm. Therefore, the TSP may serve as an appropriate benchmark case to assess how well various population seeding approaches perform.

In addition to the suggested seeding technique, we give the empirical results and analysis of the performance of the MLRBT [13] population seeding method. Table 1 shows the GA settings that were chosen for our experiments. The experiment was carried out ten times for each instance.

Table 1. GA setup parameters.

Parameter	Value/Method
Population size	100
Generation limit	3000
Initialization method	IAMTSP method and Multi-Linear Regression-Based Technique (MLRBT) [13]
Crossover method	One-point modified
Crossover rate	82%
Mutation	Exchange mutation
Mutation rate	100%
Selection mechanism	Truncation Selection
Termination criterion	Generation limit

Each individual's fitness level was determined using a truncation selection approach. This is a typical method for assigning the fitness function to every chromosome in the GA population, and truncation selection is the most straightforward method of selection. In this type of selection, the population is sorted by fitness, and then the lower-fitness proportion is dropped [51].

For the reproduction process in our experiments, the one-point modified crossover and exchange mutation methods were used, as they are among the simplest methods that have been used with problems that are characterized as permutation problems. The one-point modified crossover process randomly selects a location in the chromosomes, and then the individuals exchange genes to make new offspring [52]. The exchange mutation process works by randomly selecting two genes and altering their locations [53].

We would like to clarify that the primary objective of our research is to propose a new method aimed at enhancing diversity within genetic algorithms. We employ the TSP as an example of an optimization problem to facilitate a comparative analysis of our proposed method against other existing methods. Additionally, our choice of straightforward GA parameters, as depicted in Table 1, was intentional. We did not employ sophisticated parameter control or tuning procedures. This approach aligns with the primary goal of our paper, which is to validate the effectiveness and showcase the benefits of our proposed method when compared to a GA in terms of time and diversity. This choice of parameters was made to ensure that the results obtained were a direct reflection of the proposed method's efficacy, rather than the impact of parameter tuning.

4.1. Experimental Setup

To ensure a fair assessment, all tests were conducted in the same environment for all initialization methods. The findings aid in evaluating the effectiveness and efficiency of the IAM-TSP+ for GA population initialization when compared to other initialization methods. Simulation experiments were performed in the Microsoft Visual Studio 2022 environment, and the system's hardware and software specifications are as follows:

- 11th Gen Intel(R) Core (TM) i7-1165G7 @ 80 GHz 2.80 GHz;
- 8.00 GB of RAM;
- Windows 11 Pro, 64-bit operating system.

We conducted tests on many TSPs, each having a known optimal solution. These TSPs are from the TSPLIB [48], which includes vertices ranging from 48 to 783, and are as follows: KroA100, eil51, bier127, pr439, KroA200, lin318, pr144, att532, rat783, a280, and att48. The experiments included using the IAMTSP+ seeding method alongside the Multi-Linear Regression-Based Technique (MLRBT) [13], which was proven to be superior to the Regression-Based Technique (RBT) [22], as well as the random and NN approaches.

4.2. Experimental Results and Discussion

The experiment was carried out ten times for each instance, using the same set of parameters for the two seeding methods: the MLRBT [13] and the IAMTSP+ seeding method. The IAMTSP+ seeding method was found to be more efficient than the MLRBT [13] across all instances. Additionally, Table 2 and Figure 4 demonstrate that, in terms of the optimal solution, the IAMTSP+ technique is superior in every case. Moreover, it can be said that this approach is more effective than random, NN, and RBT [22] methods, as the suggested technique achieved good performance and produced a result that was close to the optimum.

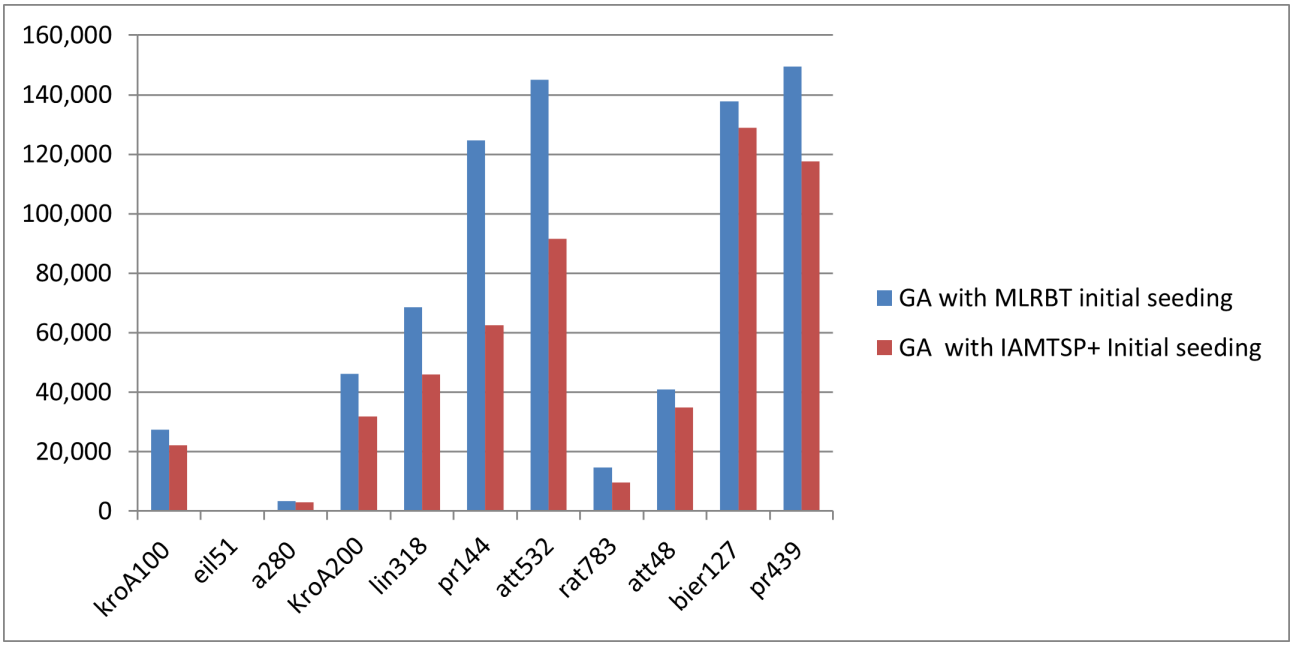

Figure 4. The results of the initial population methods on the genetic algorithm.

Table 2. The experimental results.

Instance	Optimal Solution	MLRBT		IAMTSP+	
		Best Solution	Average	Best Solution	Average
kroA100	21,282	27,493	29,440.8	22,075	22,067.4
eil51	426	465	478.6	437	437
a280	2579	3473	3539.7	2957	2964.5
KroA200	29,368	46,269	47,877.4	31,788	31,818.3
lin318	42,029	68,490	70,237.6	46,045	46,135.7
pr144	58,537	124,763	131,974.3	62,446	62,446
att532	27,686	145,128	157,423.6	91,627	91,992.4
rat783	8806	14,659	15,308.1	9725	9737
att48	10,628	40,939	41,236.9	34,877	34,877
bier127	118,282	137,850	141,773.7	128,848	128,848
pr439	107,217	149,445	154,323.3	117,650	118,096.7

When investigating various initialization methods, parameters or factors related to performance (which have been recognized as measurements), such as the error percentage, average convergence rate, and convergence diversity, must be considered. To obtain a nearly optimal solution, these factors are used to evaluate the quality of the acquired population by assessing the effect of the initial population approach on GA performance. For any given problem, the error rate is the percentage difference between the value of the solution's fitness and the known optimal solution [23]. The error rate can be computed using Equation (1). The average convergence rate of a solution is defined as the percentage of fitness reached by the solution considering the known optimal solution to the problem [32]. It can be stated as in Equation (2).

$$Error\ Rate = \frac{Fitness - Optimal\ fitness}{fitness} \times 100\% \quad (1)$$

$$Average\ Convergence = 1 - \frac{Average\ Fitness - Optimal\ fitness}{Average\ Fitness} \times 100\% \quad (2)$$

The experimental results of the error rate are given in Table 3, which includes the error rate of the best solution for every instance. Table 3 shows that the suggested approach produced the lowest possible error rate. Therefore, the results indicate that the individuals generated by the IAMTSP+ method for GAs fit the quality metrics more closely than the individuals generated by the MLRBT.

It is noteworthy that the error rates for instances att532 and att48 are substantial for all methods. The best-known solutions for both cases are substantially higher, with 86,729 and 33,522 route lengths, respectively, according to recent research [54–56]. Therefore, if we adopted these values as the real optimal solutions, the error rates would be relatively small, 5.3% and 3.9% for att532 and att48, respectively. Since we wanted to be safe, we went with the smallest optimal solutions reported, regardless of their correctness, as stated in [57–59]. It is important to note, nevertheless, that these chosen numbers seem to be far lower than the generally agreed-upon values using Euclidean distances. It is possible that measurements other than the Euclidean distance were used to calculate the reported route lengths for these two instances. However, investigating the correctness of the optimal solutions for these two instances is beyond the scope of this paper.

Table 3. Error percentage results for the two seeding methods.

Instance	MLRBT	IAMTSP+
kroA100	22.59121	3.592298981
eil51	8.387097	2.517162471
a280	25.74143	12.78322624
KroA200	36.5277	7.612935699
lin318	38.63484	8.721902487
pr144	53.08144	6.259808475
att532	80.92305	69.78401563
rat783	39.92769	9.449871465
att48	74.03942	69.52719557
bier127	14.19514	8.200360114
pr439	28.25655	8.867828304

Figure 5 shows the error rates of the initial population methods for all instances.

Table 4 and Figure 6 show the average convergence of the two population initialization methods and show how efficient the genetic algorithm is in converging to the optimal solution. As shown in Figure 6, the IAMTSP+ method achieved a larger convergence rate than the MLRBT.

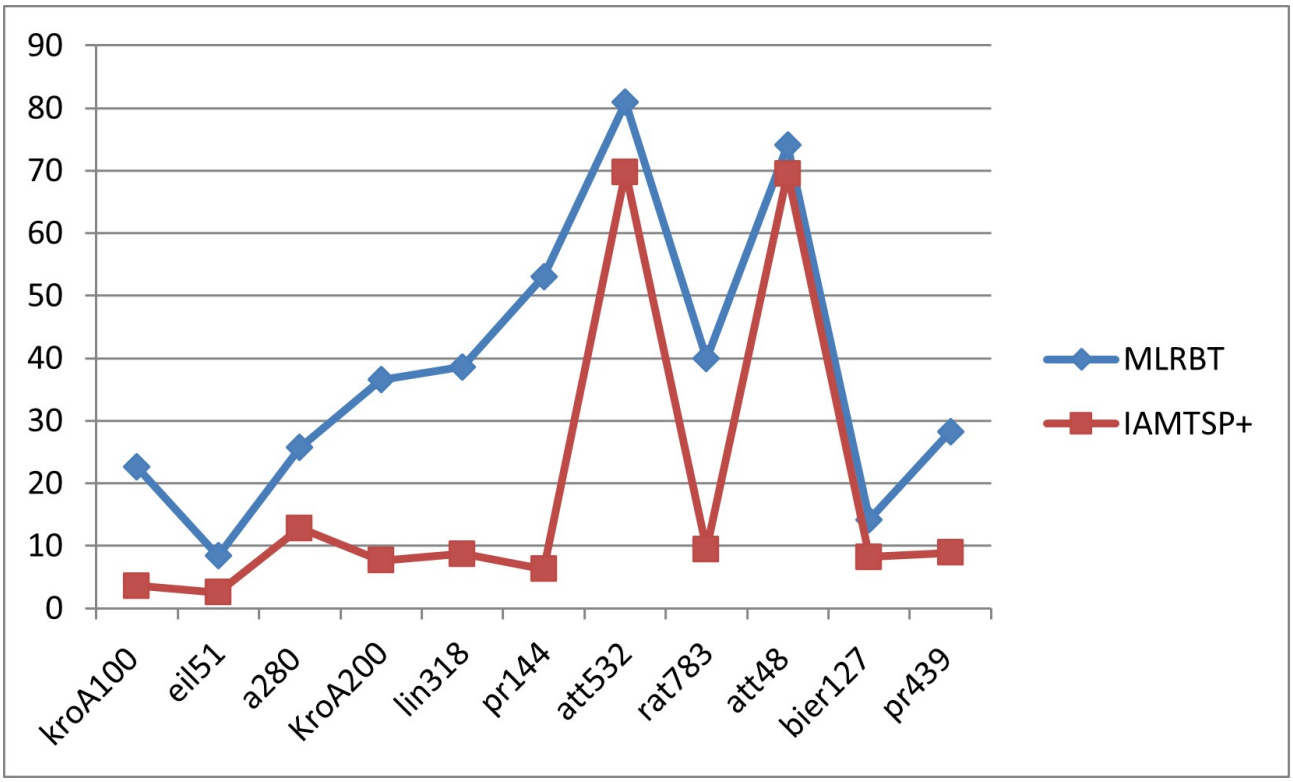

Figure 5. Error rates of population seeding methods.

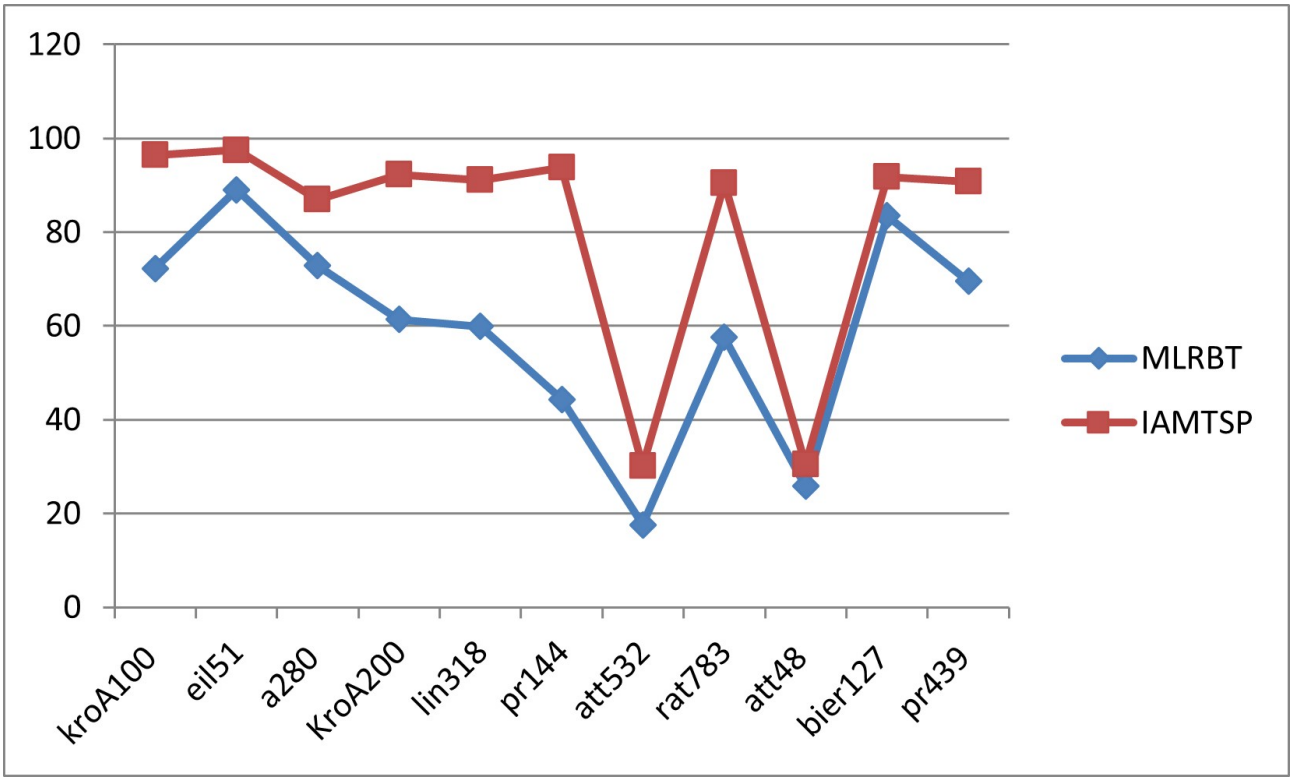

Figure 6. Average convergence rates of population seeding methods.

Table 4. Average convergence rates %.

Instance	MLRBT	IAMTSP+
kroA100	72.28744	96.44090378
eil51	89.00961	97.48283753
a280	72.85928	86.99612076
KroA200	61.34001	92.29908575
lin318	59.83832	91.09865029
pr144	44.35485	93.74019153
att532	17.58694	30.09596445
rat783	57.5251	90.43853343
att48	25.77303	30.47280443
bier127	83.43014	91.79963989
pr439	69.47557	90.78746485

The GA efficiency is significantly affected by the genetic operator choices (such as selection, crossover, and mutation), as well as the fine-tuning of the control parameters and the population seeding methods. To obtain near-optimal solutions in an appropriate amount of time, it is important to carefully select GA operators and parameters that balance exploitation and exploration and another parameter, such as the initial population method. Exploration is focused on discovering new and unknown places in the search space, whereas exploitation uses the knowledge gained from previously visited sites to identify better points. Research on GAs focuses on the selection of genetic operators and parameters and their impact on algorithm efficiency. There is no "best" answer to these challenges, and the choice of parameters is determined by the problem domain and the structure of the search space.

To examine the efficiency of the IAMTSP+ initial population seeding method and its impact on the efficiency of GA, the algorithm was run for a specific time (Millisecond) with the three population seeding methods (IAMTSP+, random, and MLRBT) for each city, and the solutions were obtained. It is worth noting that the GA parameters are the same as those displayed in Table 1, except that the termination criterion was set to a specific time for each city.

The time was first calculated for the IAMTSP+ seeding population while filling in the population size for city N, so it was time T1. Then, a specific time was fixed for the genetic algorithm that works after seeding, so it was time T2. The genetic algorithm was run with the random seeding of the population at time $T1 + T2$.

Table 5 and Figure 7 show the computational results of the genetic algorithm using the IAMTSP+ initial population seeding method compared to two other methods of initial population seeding: random and MLRBT methods.

It is clear from Table 5 that the quality of the resulting solutions for the IAMTSP+ initial population seeding method is the highest, despite the limited time, and it is also clear that the IAMTSP+ seeding method is superior to the other methods from an algorithm efficiency aspect. It is also obvious from Table 5 how the GA proceeds with the doubling of time, as despite the doubling of time for the genetic algorithm that uses random seeding, a local optimum was obtained, and the solutions obtained from the GA with the IAMTSP+ initial population method are still much better than the GA with random seeding solutions, and this justifies our use of the IAMTSP+ seeding method with the GA.

Table 5. Computational results for the GA with a specific amount of time for each instance.

Instance	Time (ms)	Time ∗ 2 (ms)	GA Solution with Random Seeding	GA Solution with Random Seeding with Time ∗ 2	GA Solution with Initial IAMTSP+	GA Solution with Initial Regression Based
bier127	6023	12,046	214,928	187,452	128,678	139,460
a280	19,399	38,798	8243	7148	3018	4525
kroA100	5565	11,130	41,233	34,176	22,057	26,040
kroA200	9403	18,806	96,272	75,012	32,088	43,140
lin318	27,772	55,544	202,537	139,552	47,507	63,085
pr144	6618	13,236	193,568	130,639	62,481	95,713
att532	173,946	347,892	358,763	348,633	96,050	126,337
att48	5136	10,272	46,780	40,711	34,877	38,544
rat783	803,368	1,606,736	51,239	49,520	10,196	11,999
pr439	89,720	179,440	380,537	390,937	122,586	166,083
eil51	5151	10,302	493	477	438	463

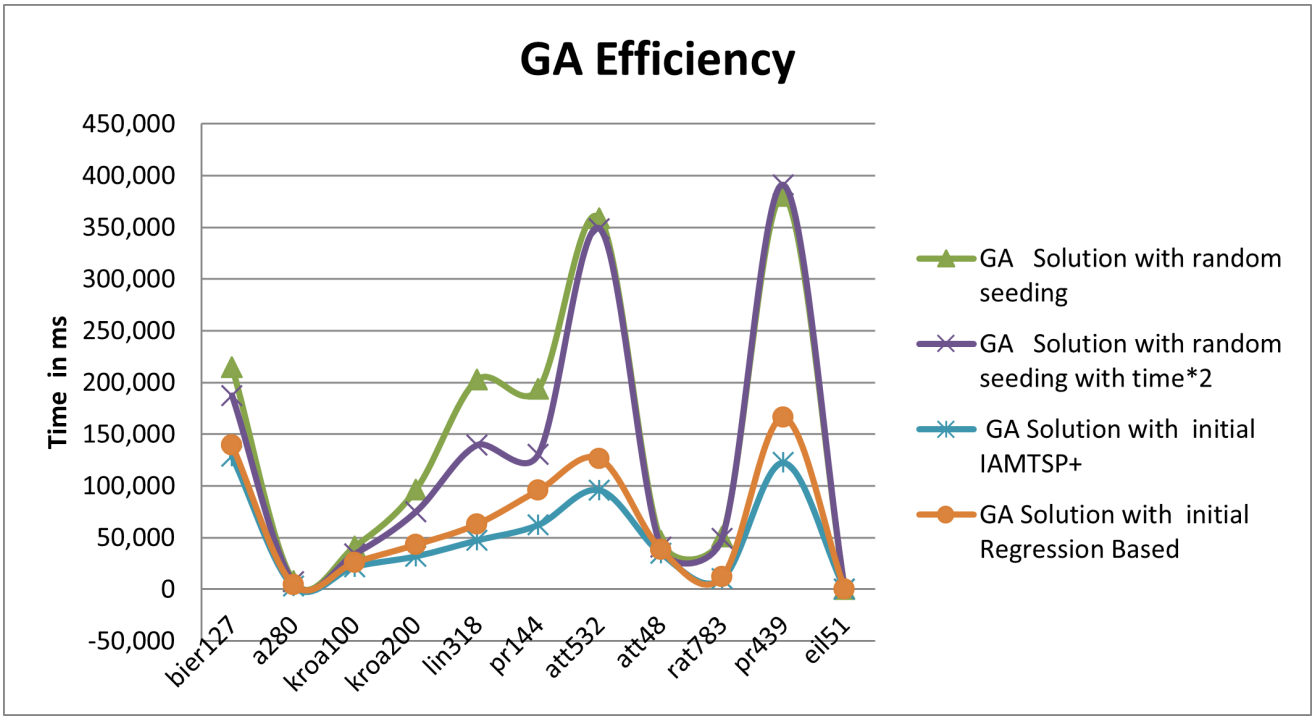

Figure 7. Comparison of the final solution in a specific amount of time.

Figure 8 indicates that the GA with IAMTSP+ initial population seeding reaches the near-optimal solution faster than the GA with random seeding, and the specified time to complete the algorithm for the two methods is 6023 ms.

Efficiency is a general term that, depending on the situation, can be defined and assessed in several ways. It frequently entails using the fewest resources—such as money, time, or energy—to achieve the intended result. When addressing problems or coming up with solutions, we typically take the amount of time needed to determine the efficiency of a particular approach. Efficiency, for instance, could be quantified as the ratio of the input—in this case, time—used to produce the output, which is the quality of the solution [16].

$$Efficiency\ (A) = \frac{Quality\ of\ Solution\ (A)}{Time\ consumed\ (A)} \quad (3)$$

where A is a problem-solving method, i.e., TSP, and the Quality of the Solution can be defined using Equation (2) (average convergence rate.) This allows the definition of efficiency for a TSP solver to be [16]

$$Efficiency\ (A) = \frac{Convergence\ Rate\ (A)}{Time\ consumed\ (A)} \tag{4}$$

This formula implies that a more efficient solution achieves a high-quality result using fewer resources (time).

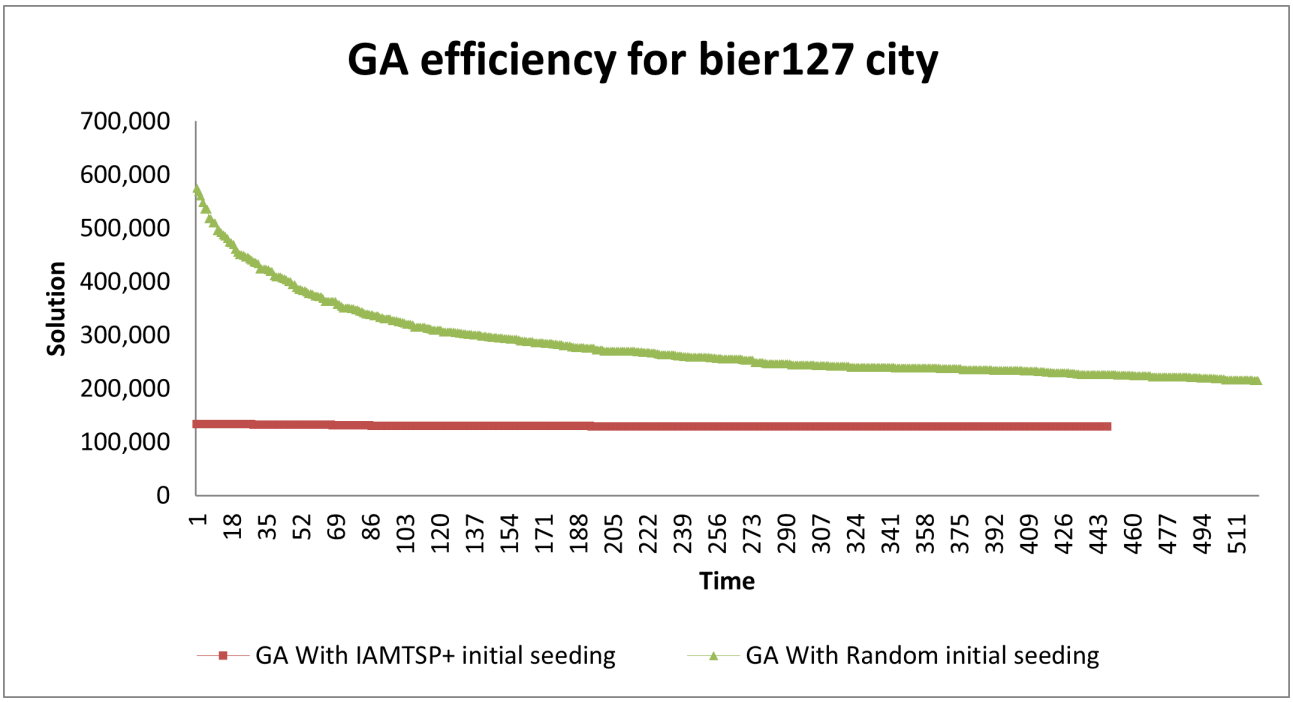

Figure 8. The efficiency of the GA for bier127 from the TSPLIB using two population seeding methods: random and IAMTSP+.

Table 6 and Figure 9 show the calculation of the error rate for each initial seeding method, but the termination criterion was running to the specified time.

Table 6. The error rate for the GA with 3 initial seeding methods based on the specified time.

Instance	GA with Random Seeding	GA with Random Seeding (Time ∗ 2)	GA with IAMTSP+ Seeding	GA with Regression-Based Seeding
bier127	44.96669	36.9001131	8.0790811	15.18572
a280	68.71285	63.91997762	14.546057	43.00552
kroA100	48.386	37.72823034	3.5136238	18.27189
kroA200	69.49476	60.84893084	8.4766891	31.92397
lin318	79.24873	69.88291103	11.530932	33.37719
pr144	69.75895	55.19178806	6.3123189	38.84112
att532	92.28293	92.05869783	71.175429	78.0856
att48	77.28089	73.89403355	69.527196	72.42632
rat783	82.81387	82.21728595	13.632797	26.61055
pr439	71.82482	72.57435341	12.537321	35.44372
eil51	13.59026	10.6918239	2.739726	7.991361

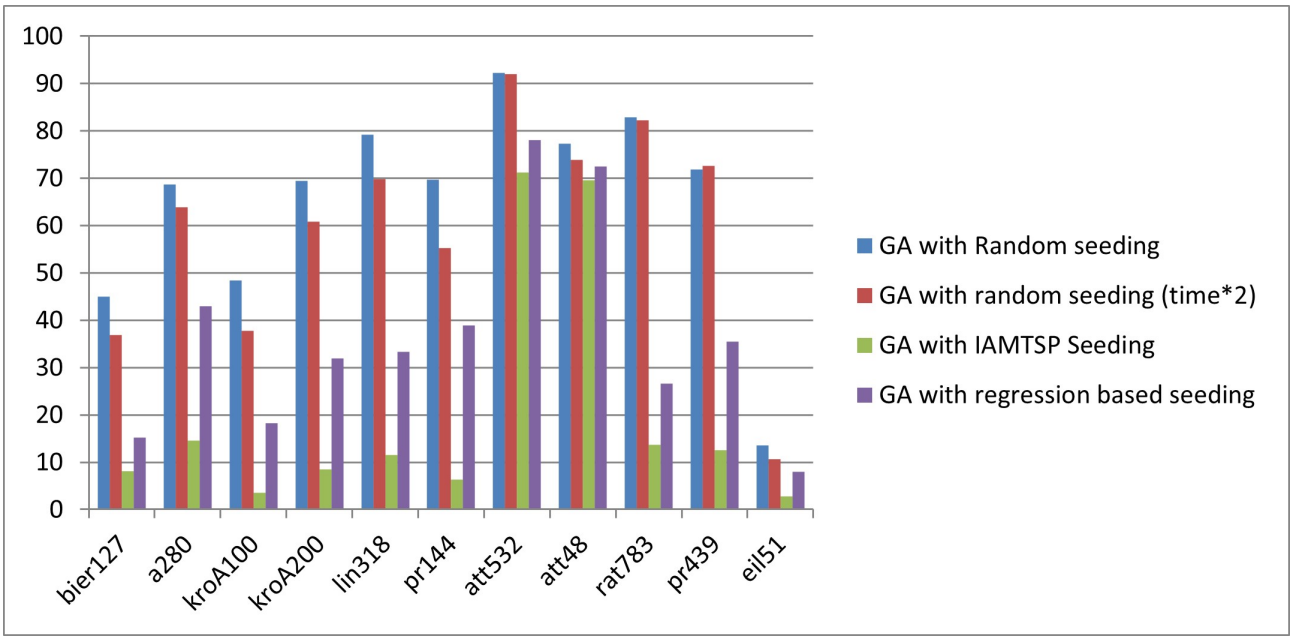

Figure 9. The error rate of each initial seeding method based on specific time.

Based on Equation (3), Table 7 displays the GA efficiency for each instance, as well as the methods used to seed the initial population. From Table 7, we conclude that employing IAMTSP+ as an initial population seeding method with the GA improves the efficiency of the GA when solving the TSP.

Table 7. GA efficiency.

Instance	GA Solution with Random Seeding	GA Solution with Random Seeding with Time ∗ 2	GA Solution with Initial IAMTSP+	GA Solution with Regression Based
bier127	0.009137193	0.005238244	0.01526165	0.014081734
a280	0.001612823	0.000929945	0.004405069	0.002938011
kroA100	0.009274753	0.005594948	0.017338073	0.014686094
kroA200	0.003244202	0.002081839	0.009733416	0.00723982
lin318	0.000747201	0.00054222	0.003185549	0.00239892
pr144	0.004569515	0.003385329	0.014156495	0.009241294
att532	0.0000443648	0.0000228269	0.00016571	0.000125984
att48	0.004423503	0.002541469	0.005933178	0.005368708
rat783	0.0000213926	0.0000110676	0.000107506	0.0000913522
pr439	0.000314035	0.00015284	0.00097484	0.00071953

Table 7 demonstrates that the proposed GA-IAMTSP+ outperforms both the pure GA and the GA that initially began with a regression approach in terms of efficiency across all TSP instances. This notable achievement is attributable to the high-quality initial solutions offered by IAMTSP+. Because these initial solutions are of greater quality, the GA converges faster and produces solutions of higher quality. This effectively addresses our study's main concern: how can the GA's efficiency in solving the TSP be improved?

4.3. Experiments on Simulated Data

We conducted three experiments on randomly generated instances, C1, C2, and C3, with 100, 200, and 300 cities, respectively. We used a uniform distribution to simulate city locations (x- and y-coordinates), ensuring that cities were randomly but uniformly dispersed over the simulation space. These experiments utilized a variety of seeding approaches, including random seeding, MLRBT seeding, and IAMTSP+ seeding, while

keeping GA parameters constant, as shown in Table 1. Table 8 shows the lengths of the routes obtained by the GA with different seeding approaches.

As can be seen from Table 8 and Figure 10, it appears that the IAMTSP+ seeding method consistently outperforms the other methods across all instances. Specifically, using the GA with IAMTSP+ seeding resulted in the shortest routes for all three instances. This finding aligns with outcomes previously observed in real-world TSP instances, suggesting that adopting a more sophisticated and adaptive approach for the initial population in the GA culminates in the enhanced performance of the GA's final solution.

By demonstrating the superior performance of the IAMTSP+ method on real-world and simulated data, we can build confidence in its use for generating the initial seeding for the GA to be used not only for solving the classical TSP but also in various other applications, such as networking [60,61], transportation planning, urban design, location-based services, etc.

Table 8. Comparison of route length results obtained by the GA on simulated data utilizing three seeding approaches, bold font means the best performance.

Instance	Number of Cities	MLRBT	IAMTSP+	GA with Random	GA with MLRBT	GA with IAMTSP+
C1	100	3484	3175	3813	3220	**3142**
C2	200	6307	4514	6714	5363	**4418**
C3	300	8630	5750	12167	6510	**5623**

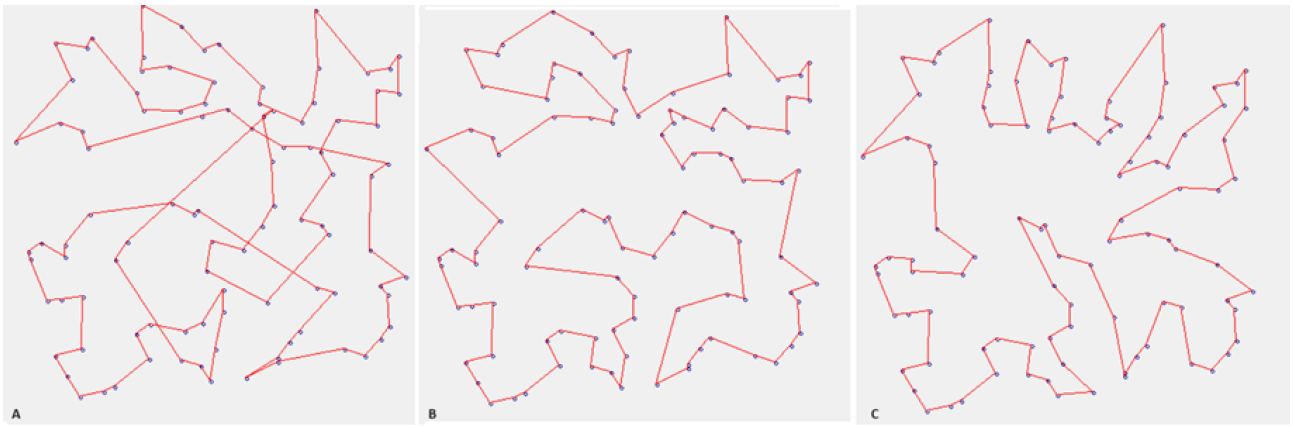

Figure 10. Visualization of the resultant routes from the GA after applying the three initial seeding methods on C1. (**A**) Random seeding, (**B**) MLRBT seeding, (**C**) IAMTSP+ seeding.

5. Conclusions

We present a GA approach with improved efficiency in this study that makes use of the IAMTSP+ initial population strategy. This approach's main benefits include a considerable reduction in processing time, increased robustness, and better solution quality. We evaluated the new method on popular TSP instances to determine its performance. When we compared our method to previously published ones in the literature, including regression-based and random initial solutions, we found that using IAMTSP+ as an initial population method for GA performed better than the other methods in all TSP instances.

Our goal in this work was to investigate the effect of different initiation procedures on the effectiveness of genetic algorithms (GAs). It is important to note, however, that the efficiency of the GA is impacted by a variety of circumstances. In future studies, we plan to thoroughly examine the IAMTSP+ technique by taking into account other factors, such as selection methods, deriving the remaining solutions from a single IAMTSP+ solution, and combining two initialization procedures to seed the population. We want to conduct thorough testing and comparisons in our next study, with the goal of providing a detailed evaluation of the effectiveness of the proposed GA-IAMTSP+ approach.

Author Contributions: Conceptualization, E.A. and A.H.; methodology, A.H., E.A. and S.E.; software, A.H. and S.E.; validation, E.A. and E.E.; formal analysis, E.A. and E.E.; investigation, E.E. and A.H.; resources, E.A. and E.E.; data curation, S.E. and E.A.; writing—original draft preparation, A.H., E.A., S.E. and E.E.; writing—review and editing, A.H., E.A., S.E. and E.E.; visualization, E.E. and E.A.; supervision, A.H. and S.E.; project administration, S.E. and E.E. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data that support the findings of this study are openly available in TSPLIB at http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/ (accessed on 22 January 2024).

Acknowledgments: We genuinely appreciate the reviewers' voluntary efforts and are grateful for their valuable insights.

Conflicts of Interest: The authors have no conflicts of interest to declare. All co-authors have seen and agree with the contents of the manuscript, and there are no financial interests to report. We certify that the submission is original work and is not under review at any other publication.

Abbreviations

The following abbreviations are used in this manuscript:

GA	Genetic algorithm
NN	Nearest neighbor
TSP	Traveling Salesman Problem
IAM-TSP	Iterative Approximate Methods
MLRBT	Multi-Linear Regression-Based Technique

References

1. Zhou, G.; Zhu, Z.; Luo, S. Location optimization of electric vehicle charging stations: Based on cost model and genetic algorithm. *Energy* **2022**, *247*, 123437. [CrossRef]
2. Han, S.; Xiao, L. An improved adaptive genetic algorithm. *Proc. Shs Web Conf. Edp Sci.* **2022**, *140*, 01044. [CrossRef]
3. Bi, H.; Lu, F.; Duan, S.; Huang, M.; Zhu, J.; Liu, M. Two-level principal–agent model for schedule risk control of IT outsourcing project based on genetic algorithm. *Eng. Appl. Artif. Intell.* **2020**, *91*, 103584. [CrossRef]
4. Arram, A.; Ayob, M. A novel multi-parent order crossover in genetic algorithm for combinatorial optimization problems. *Comput. Ind. Eng.* **2019**, *133*, 267–274. [CrossRef]
5. Hassanat, A.B.; Alkafaween, E.; Al-Nawaiseh, N.A.; Abbadi, M.A.; Alkasassbeh, M.; Alhasanat, M.B. Enhancing genetic algorithms using multi mutations: Experimental results on the travelling salesman problem. *Int. J. Comput. Sci. Inf. Secur.* **2016**, *14*, 785.
6. Lu, F.; Bi, H.; Huang, M.; Duan, S. Simulated annealing genetic algorithm based schedule risk management of IT outsourcing project. *Math. Probl. Eng.* **2017**, *2017*, 6916575. [CrossRef]
7. Hassanat, A.; Almohammadi, K.; Alkafaween, E.; Abunawas, E.; Hammouri, A.; Prasath, V.S. Choosing mutation and crossover ratios for genetic algorithms—A review with a new dynamic approach. *Information* **2019**, *10*, 390. [CrossRef]
8. Paul, P.V.; Ramalingam, A.; Baskaran, R.; Dhavachelvan, P.; Vivekanandan, K.; Subramanian, R. A new population seeding technique for permutation-coded Genetic Algorithm: Service transfer approach. *J. Comput. Sci.* **2014**, *5*, 277–297. [CrossRef]
9. Shanmugam, M.; Basha, M.S.; Paul, P.V.; Dhavachelvan, P.; Baskaran, R. Performance assessment over heuristic population seeding techniques of genetic algorithm: Benchmark analyses on traveling salesman problems. *Int. J. Appl. Eng. Res. (Ijaer) Res. India Publ.* **2013**, *8*, 1171–1184.
10. Riazi, A. Genetic algorithm and a double-chromosome implementation to the traveling salesman problem. *Appl. Sci.* **2019**, *1*, 1397. [CrossRef]
11. Hassanat, A.B.; Alkafaween, E. On enhancing genetic algorithms using new crossovers. *Int. J. Comput. Appl. Technol.* **2017**, *55*, 202–212. [CrossRef]
12. Katoch, S.; Chauhan, S.S.; Kumar, V. A review on genetic algorithm: Past, present, and future. *Multimed. Tools Appl.* **2021**, *80*, 8091–8126. [CrossRef]
13. Alkafaween, E.; Hassanat, A.B.; Tarawneh, S. Improving initial population for genetic algorithm using the multi linear regression based technique (MLRBT). *Commun.-Sci. Lett. Univ. Zilina* **2021**, *23*, E1–E10. [CrossRef]

14. Alkafaween, E.; Hassanat, A. Improving TSP Solutions Using GA with a New Hybrid Mutation Based on Knowledge and Randomness. *Komunikácie* **2020**, *22*, 12. [CrossRef]
15. Bhandari, A.; Tripathy, B.; Jawad, K.; Bhatia, S.; Rahmani, M.K.I.; Mashat, A. Cancer detection and prediction using genetic algorithms. *Comput. Intell. Neurosci.* **2022**, *2022*, 1871841. [CrossRef] [PubMed]
16. Hassanat, A. Greedy algorithms for approximating the diameter of machine learning datasets in multidimensional Euclidean space: Experimental results. *Adcaij Adv. Distrib. Comput. Artif. Intell. J.* **2018**, *7*, 15. [CrossRef]
17. Paul, V.; Ganeshkumar, C.; Jayakumar, L. Performance evaluation of population seeding techniques of permutation-coded GA traveling salesman problems based assessment: Performance evaluation of population seeding techniques of permutation-coded GA. *Int. J. Appl. Metaheuristic Comput. (Ijamc)* **2019**, *10*, 55–92. [CrossRef]
18. Toğan, V.; Daloğlu, A.T. An improved genetic algorithm with initial population strategy and self-adaptive member grouping. *Comput. Struct.* **2008**, *86*, 1204–1218. [CrossRef]
19. Pan, W.; Li, K.; Wang, M.; Wang, J.; Jiang, B. Adaptive randomness: A new population initialization method. *Math. Probl. Eng.* **2014**, *2014*, 975916. [CrossRef]
20. Maaranen, H.; Miettinen, K.; Penttinen, A. On initial populations of a genetic algorithm for continuous optimization problems. *J. Glob. Optim.* **2007**, *37*, 405–436. [CrossRef]
21. Keedwell, E.; Khu, S.T. A hybrid genetic algorithm for the design of water distribution networks. *Eng. Appl. Artif. Intell.* **2005**, *18*, 461–472. [CrossRef]
22. Hassanat, A.B.; Prasath, V.S.; Abbadi, M.A.; Abu-Qdari, S.A.; Faris, H. An improved genetic algorithm with a new initialization mechanism based on regression techniques. *Information* **2018**, *9*, 167. [CrossRef]
23. Ray, S.S.; Bandyopadhyay, S.; Pal, S.K. Genetic operators for combinatorial optimization in TSP and microarray gene ordering. *Appl. Intell.* **2007**, *26*, 183–195. [CrossRef]
24. Yang, R. Solving large travelling salesman problems with small populations. In Proceedings of the Second International Conference On Genetic Algorithms In Engineering Systems: Innovations and Applications, IET, Glasgow, UK, 2–4 September 1997; pp. 157–162.
25. Wei, Y.; Hu, Y.; Gu, K. Parallel search strategies for TSPs using a greedy genetic algorithm. In Proceedings of the Third International Conference on Natural Computation (ICNC 2007), Haikou, China, 24–27 August 2007; Volume 3, pp. 786–790.
26. Yugay, O.; Kim, I.; Kim, B.; Ko, F.I. Hybrid genetic algorithm for solving traveling salesman problem with sorted population. In Proceedings of the 2008 Third International Conference on Convergence and Hybrid Information Technology, Busan, Republic of Korea, 1–13 November 2008; Volume 2, pp. 1024–1028.
27. Deng, Y.; Liu, Y.; Zhou, D. An improved genetic algorithm with initial population strategy for symmetric TSP. *Math. Probl. Eng.* **2015**, *2015*. [CrossRef]
28. Li, C.; Chu, X.; Chen, Y.; Xing, L. A knowledge-based technique for initializing a genetic algorithm. *J. Intell. Fuzzy Syst.* **2016**, *31*, 1145–1152. [CrossRef]
29. Laporte, G. The traveling salesman problem: An overview of exact and approximate algorithms. *Eur. J. Oper. Res.* **1992**, *59*, 231–247. [CrossRef]
30. Potvin, J.Y. Genetic algorithms for the traveling salesman problem. *Ann. Oper. Res.* **1996**, *63*, 337–370. [CrossRef]
31. Matai, R.; Singh, S.P.; Mittal, M.L. Traveling salesman problem: An overview of applications, formulations, and solution approaches. In *Traveling Salesman Problem: Theory and Applications*; BoD–Books on Demand: Norderstedt, Germany, 2010; Volume 1, pp. 1–25.
32. Paul, P.V.; Dhavachelvan, P.; Baskaran, R. A novel population initialization technique for genetic algorithm. In Proceedings of the 2013 International Conference on Circuits, Power and Computing Technologies (ICCPCT), Nagercoil, India, 20–21 March 2013; pp. 1235–1238.
33. Hoffman, K.L.; Padberg, M.; Rinaldi, G. Traveling salesman problem. *Encycl. Oper. Res. Manag. Sci.* **2013**, *1*, 1573–1578.
34. Davendra, D. *Traveling Salesman Problem: Theory and Applications*; BoD–Books on Demand: Norderstedt, Germany, 2010.
35. Lu, F.; Chen, W.; Feng, W.; Bi, H. 4PL routing problem using hybrid beetle swarm optimization. *Soft Comput.* **2023**, *27*, 17011–17024. [CrossRef]
36. Gülcü, Ş.; Mahi, M.; Baykan, Ö.K.; Kodaz, H. A parallel cooperative hybrid method based on ant colony optimization and 3-Opt algorithm for solving traveling salesman problem. *Soft Comput.* **2018**, *22*, 1669–1685. [CrossRef]
37. Feng, X.; Lau, F.C.; Gao, D. A new bio-inspired approach to the traveling salesman problem. In Proceedings of the Complex Sciences: First International Conference, Complex 2009, Shanghai, China, 23–25 February 2009; Revised Papers, Part 21; Springer: Berlin/Heidelberg, Germany, 2009; pp. 1310–1321.
38. Hao, Z.; Huang, H.; Cai, R. Bio-inspired Algorithms for TSP and Generalized TSP. In *Traveling Salesman Problem*; Greco, F., Ed.; InTech Open: London, UK, 2008; pp. 35–62, ISBN 978-953-7619-10-7.
39. Brady, R. Optimization strategies gleaned from biological evolution. *Nature* **1985**, *317*, 804–806. [CrossRef]
40. Scholz, J. Genetic algorithms and the traveling salesman problem a historical review. *arXiv* **2019**, arXiv:1901.05737.
41. Alkafaween, E.; Elmougy, S.; Essa, E.; Mnasri, S.; Tarawneh, A.S.; Hassanat, A. IAM-TSP: Iterative Approximate Methods for Solving the Travelling Salesman Problem. *Int. J. Adv. Comput. Sci. Appl.* **2023**, *14*, 11. [CrossRef]
42. Abdallah, W.; Val, T. Genetic-Voronoi algorithm for coverage of IoT data collection networks. In Proceedings of the 2020 30th International Conference on Computer Theory and Applications (ICCTA), Virtual, 12–14 December 2020; pp. 16–22.

43. Yuan, Q.; Wang, S.; Hu, M.; Zeng, L. SLDChOA: A comprehensive and competitive multi-strategy-enhanced chimp algorithm for global optimization and engineering design. *J. Supercomput.* **2024**, *80*, 3589–3643. [CrossRef]
44. Mnasri, S.; Thaljaoui, A.; Nasri, N.; Val, T. A genetic algorithm-based approach to optimize the coverage and the localization in the wireless audio-sensors networks. In Proceedings of the 2015 International Symposium on Networks, Computers and Communications (ISNCC), Hammamet, Tunisia, 13–15 May 2015; pp. 1–6.
45. Mnasri, S.; Nasri, N.; Van Den Bossche, A.; Val, T. A hybrid ant-genetic algorithm to solve a real deployment problem: A case study with experimental validation. In Proceedings of the Ad-hoc, Mobile, and Wireless Networks: 16th International Conference on Ad Hoc Networks and Wireless, ADHOC-NOW 2017, Messina, Italy, 20–22 September 2017; Proceedings 16; Springer: Berlin/Heidelberg, Germany, 2017; pp. 367–381.
46. Nagata, Y.; Soler, D. A new genetic algorithm for the asymmetric traveling salesman problem. *Expert Syst. Appl.* **2012**, *39*, 8947–8953. [CrossRef]
47. Dao, S.D.; Abhary, K.; Marian, R. An effective genetic algorithm for large-scale traveling salesman problems. In Proceedings of the World Congress on Engineering and Computer Science, San Francisco, CA, USA, 19–21 October 2016; Volume 1.
48. Reinelt, G. TSPLIB, 1996. 12 2. 2023. Available online: http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/ (accessed on 22 January 2024).
49. Sivanandam, S.; Deepa, S.; Sivanandam, S.; Deepa, S. *Genetic Algorithms*; Springer: Berlin/Heidelberg, Germany, 2008.
50. Sun, W. A novel genetic admission control for real-time multiprocessor systems. In Proceedings of the 2009 International Conference on Parallel and Distributed Computing, Applications and Technologies, Higashi Hiroshima, Japan, 8–11 December, 2009; pp. 130–137.
51. Jebari, K.; Madiafi, M. Selection methods for genetic algorithms. *Int. J. Emerg. Sci.* **2013**, *3*, 333–344.
52. Bala, A.; Sharma, A.K. A comparative study of modified crossover operators. In Proceedings of the 2015 Third International Conference on Image Information Processing (ICIIP), Waknaghat, India, 21–24 December 2015; pp. 281–284.
53. Banzhaf, W. The "molecular" traveling salesman. *Biol. Cybern.* **1990**, *64*, 7–14. [CrossRef]
54. Song, J.; Pu, Y.; Xu, X. Adaptive Ant Colony Optimization with Sub-Population and Fuzzy Logic for 3D Laser Scanning Path Planning. *Sensors* **2024**, *24*, 1098. [CrossRef]
55. Wang, J.; Zhang, P.; Zhang, H.; Song, H.; Bei, J.; Sun, W.; Sun, X. A carnivorous plant algorithm with heuristic decoding method for traveling salesman problem. *IEEE Access* **2022**, *10*, 97142–97164. [CrossRef]
56. Pan, H.; You, X.; Liu, S. High-frequency path mining-based reward and punishment mechanism for multi-colony ant colony optimization. *IEEE Access* **2020**, *8*, 155459–155476. [CrossRef]
57. Gharehchopogh, F.S.; Abdollahzadeh, B.; Arasteh, B. An improved farmland fertility algorithm with hyper-heuristic approach for solving travelling salesman problem. *Cmes-Comput. Model. Eng. Sci.* **2022**, *135*, 1–26.
58. Hussain, A.; Muhammad, Y.S.; Sajid, M.N. A simulated study of genetic algorithm with a new crossover operator using traveling salesman problem. *J. Math.* **2019**, *51*, 61–77.
59. Shahab, M. New heuristic algorithm for traveling salesman problem. *Proc. J. Phys. Conf. Ser. Iop Publ.* **2019**, *1218*, 012038. [CrossRef]
60. Btoush, A.; Tareef, A.; Alkasasbeh, A.A. Network Propagation Loss Models: Effects and Classification. In Proceedings of the 2022 International Conference on Emerging Trends in Computing and Engineering Applications (ETCEA), Karak, Jordan, 23–25 November 2022; pp. 1–6.
61. Abadleh, A.; Btoush, A.; Alkasasbeh, A.A.; Mahadeen, A.; Al-Hawari, E.; Tareef, A.; Al-Mjali, M.M. Mitigating the Effect of Blackhole Attacks in MANAT. *J. Eng. Sci. Technol. Rev.* **2022**, *15*, 107. [CrossRef]