information

Article

Choosing Mutation and Crossover Ratios for Genetic Algorithms—A Review with a New Dynamic Approach

Ahmad Hassanat [1,2,*,†,‡], **Khalid Almohammadi** [1], **Esra'a Alkafaween** [2,‡], **Eman Abunawas** [2], **Awni Hammouri** [2] **and V. B. Surya Prasath** [3,4,5,6]

1 Computer Science Department, Community College, University of Tabuk, Tabuk 71491, Saudi Arabia; kalmohammadi@ut.edu.sa
2 Computer Science Department, Mutah University, Karak 61711, Jordan; esrakafaween86@gmail.com (E.A.); emanyounes1991@gmail.com (E.A.); hammouri@mutah.edu.jo (A.H.)
3 Division of Biomedical Informatics, Cincinnati Children's Hospital Medical Center, OH 45229, USA; prasatsa@uc.edu
4 Department of Pediatrics, University of Cincinnati College of Medicine, Cincinnati, OH 45267, USA
5 Department of Biomedical Informatics, College of Medicine, University of Cincinnati, OH 45267, USA
6 Department of Electrical Engineering and Computer Science, University of Cincinnati, OH 45221, USA
* Correspondence: hasanat@mutah.edu.jo
† Current address: Computer Department, Community College, University of Tabuk, Tabuk 71491, Saudi Arabia.
‡ These authors contributed equally to this work.

Received: 6 November 2019; Accepted: 2 December 2019; Published: 10 December 2019

Abstract: Genetic algorithm (GA) is an artificial intelligence search method that uses the process of evolution and natural selection theory and is under the umbrella of evolutionary computing algorithm. It is an efficient tool for solving optimization problems. Integration among (GA) parameters is vital for successful (GA) search. Such parameters include mutation and crossover rates in addition to population that are important issues in (GA). However, each operator of GA has a special and different influence. The impact of these factors is influenced by their probabilities; it is difficult to predefine specific ratios for each parameter, particularly, mutation and crossover operators. This paper reviews various methods for choosing mutation and crossover ratios in GAs. Next, we define new deterministic control approaches for crossover and mutation rates, namely Dynamic Decreasing of high mutation ratio/dynamic increasing of low crossover ratio (DHM/ILC), and Dynamic Increasing of Low Mutation/Dynamic Decreasing of High Crossover (ILM/DHC). The dynamic nature of the proposed methods allows the ratios of both crossover and mutation operators to be changed linearly during the search progress, where (DHM/ILC) starts with 100% ratio for mutations, and 0% for crossovers. Both mutation and crossover ratios start to decrease and increase, respectively. By the end of the search process, the ratios will be 0% for mutations and 100% for crossovers. (ILM/DHC) worked the same but the other way around. The proposed approach was compared with two parameters tuning methods (predefined), namely fifty-fifty crossover/mutation ratios, and the most common approach that uses static ratios such as (0.03) mutation rates and (0.9) crossover rates. The experiments were conducted on ten Traveling Salesman Problems (TSP). The experiments showed the effectiveness of the proposed (DHM/ILC) when dealing with small population size, while the proposed (ILM/DHC) was found to be more effective when using large population size. In fact, both proposed dynamic methods outperformed the predefined methods compared in most cases tested.

Keywords: genetic algorithms; crossover; mutation; ratios; parameter selection

1. Introduction

Genetic algorithms (GAs) are adaptive meta heuristic search algorithms which classified as an evolutionary computing algorithms, which use techniques inspired by natural evolution. The first GA was developed by Holland in 1975, to solve some optimization problems, based on biological genetic and evolutionary ideas [1]. GA is a discrete and non-linear process that is not a mathematically guided algorithm where optima evolved from one generation to another without mathematical formulation [2]. The GA emerged as an important tool aiming to solve problems with multiple solutions and complex search spaces, which made it possible to solve many problems, reach several solution structures and provide solutions for parameter problems [3]. GAs have become a leading used approach to provide solutions to several complex optimization problems [4]. Accordingly, they are considered to be optimization tools [3,5]. In essence, they are widely used by researchers in many areas such as computer networks [6], software engineering [7], image processing [8,9], speech recognition [10,11], sensor networks [12,13], healthcare [14,15], computer games [16], machine learning [17] etc.

Traditional GA steps begin with the creation of a group of individuals randomly chosen to serve as a solution translated from natural language to GA units which are called the chromosomes [18]. Consequently, GA could solve any optimization problem which could be presented with the chromosomes encoding [19]. GA works by creating initial solutions (individuals). Then the algorithm works to apply an evaluation function, which is provided by the programmer and depends on the problem type. This process is basically done to evaluate the individual's goodness on how well they perform at a given problem task. After the evaluation process, two individuals are selected upon their fitness value (the goodness) [20,21]. These two individuals reproduce using a GA parameter to give one or more offspring. Each round of these processes is called a generation. This step continues until an optimal or the closest to optimal solution is found or some termination criteria are satisfied, though this depends mainly on the programmer in the first place [22].

1.1. Contributions

The effectiveness of the GAs relays on the selection of its control parameters (population size, crossover, and mutation) that interact in a complex way [23,24]. Many researchers studied the performance of the crossover and mutation operators on the effectiveness of the GAs, and whether the strength belongs to both, or in each one that used alone, as in [25,26].

In this work, we review of the literature on various choices available for choosing mutation and crossover ratios in GAs. Furthermore, we propose new deterministic control methods for crossover and mutation rates namely: Dynamic Decreasing of high mutation ratio/dynamic increasing of low crossover ratio (DHM/ILC), and Dynamic Increasing of Low Mutation/Dynamic Decreasing of High Crossover (ILM/DHC).

The dynamic nature of the proposed methods allows the ratios of both crossover and mutation operators to be changed linearly during the search progress where (DHM/ILC) started with 100% ratio for mutations, and 0% for crossovers. Both mutation and crossover ratios started to decrease and increase, respectively. By the end of the search process, the ratios will be 0% for mutations and 100% for crossovers. (ILM/DHC) worked the same but the other way around.

The proposed methods were compared with two parameters tuning methods (predefined) namely: fifty-fifty crossover/mutation ratios, and the well-known method that used common ratios with (0.03) mutation rates and (0.9) crossover rates.

1.2. Organization

We start with a review of GA representations in Section 2. The rest of this work is dedicated to presenting a review of the related works to the subject in Section 3, and we present our proposed dynamic approach in Section 4. In Section 5 we present and discuss the experimental results of the proposed approach compared with other methods, along with time complexity. Finally, Section 6 concludes this paper with future works.

2. Review of GA Representations

Figure 1 shows a typical GA as described in [27]. The common problem-dependent constituents of genetic algorithm include the following: the encoding and the evaluation functions. The researcher started with problem encoding (representation of the problem) as a first step to translate the problem to computer language [21]. The nature of the problem to be solved usually dictates the representation of the problem. There are various representation methods such as:

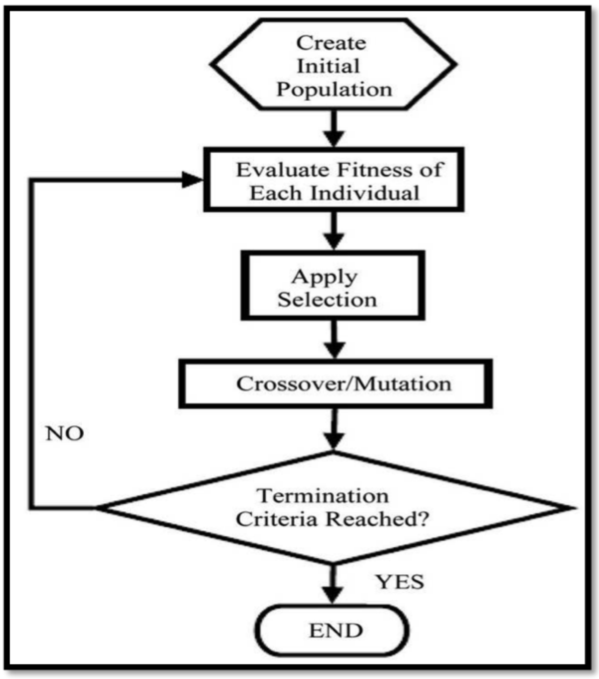

Create Initial Population
Evaluate Fitness of Each Individual
Apply Selection
Crossover/Mutation
NO
Termination Criteria Reached?
YES
END

Figure 1. Flow chart of a typical GA [27].

1. Binary encoding: Each chromosome in this technique is represented using strings of bits 0's and 1's. An example of the use of binary encoding is the knapsack problem.
2. Permutation encoding: Each chromosome in this technique is represented as a string of numbers that represent a position in a sequence. This method is useful in ordering problems such as Traveling Salesman Problem (TSP).
3. Value encoding: chromosomes are represented by using a sequence of some values such as real numbers, characters or objects. These values are possibly characters, real numbers, etc. [28].
4. Tree encoding: Each chromosome is treated as a tree of certain items such as commands or functions. Tree encoding is good for evolving programs [28] used in genetic programming. The following outline summarizes how the GA works after the chromosomes are encoded [29].

    1. Start: Generation of the beginning population of random individuals is the first step of any genetic algorithms. Each generated individual is then represented as a chromosome in a sequence of string with the length L, that aligns with the problem encoding. The step ends with creating a random population in "genotype" [1].
    2. Fitness: Next is to compute the fitness value $f(x)$ of each individual in the present population. The evaluation process consists of choosing individuals for mating based on their fitness value (parents) and according to the desired values of each. The terms evaluation and fitness are usually used to mean the same thing but it is important

to differentiate the evaluating and the fitness valuing as applied in GA. Evaluation function or objective function rates performance based on particular defined aspects. The performance measurement of the fitness function is transformed into a range of reproductive opportunities. [30].

3. New population: Steps 4, 5 and 6 are repeated to create a new population up to completion.
4. Selection (Reproduction): The selection process ascertains the chromosomes that are chosen for mating and reproduction as well as the number of offspring each chosen chromosome produces. The main purpose of the selection process is "the better an individual is; the higher its chance of being a parent" [31]. There exist various traditional selection mechanisms and user-specified-selection mechanisms that are applied in line with the problem in question. [32]. Selection strategies examples consist of:

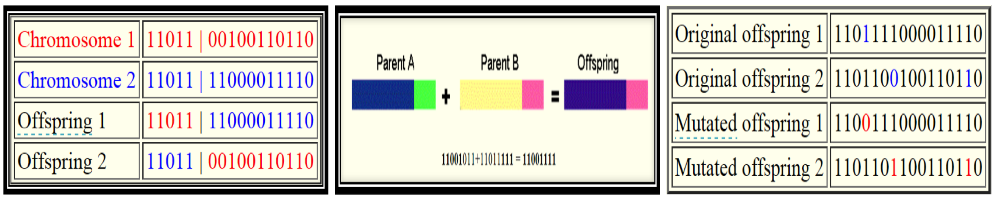
   (a) Tournament Selection: This is arguably the most common selection technique in GA because of its efficiency and ease in implementation [33]. In tournament selection, the selection of individuals is done in a random manner starting with the larger population. Thereafter there is a competition amongst the selected individuals. The competition is used to determine the individual with the highest fitness value to be used in the generation of the new population. The individuals competing are usually set to two also called binary tournament or tournament size. Diversity is ensured in tournament selection as all the individuals have an equal chance to be chosen even though this may reduce the convergence speed. Some of tournament selection advantage includes proper time utilization mostly when the implementation is parallel with low susceptibility to being taken by dominant individuals, and no need for fitness scaling or sorting. [34].

   (b) Proportional Roulette Wheel Selection: Also called fitness proportionate. Fitness values are used to choose Parents where the best chromosome stands a higher chance of selection [35,36].

   (c) Rank Selection: Parent selection is done using a rank. Here, fitness value is used to rank individuals in the population where the best is ranked (n) and the worst-ranked (1). Every chromosome has a rank depending on its expected value [31].

5. Crossover: The use of the selection process determines the parents used in the crossover to produce a new offspring. Crossover is implemented by selecting a random point on the chromosome where the parents' parts exchange happens. The crossover then brings up a new offspring based on the exchange point chosen with particular parts of the parents.

   Figure 2a depicts a crossover process [37].



Chromosome 1	11011 | 00100110110
Chromosome 2	11011 | 11000011110
Offspring 1	11011 | 11000011110
Offspring 2	11011 | 00100110110

Original offspring 1	1101111000011110
Original offspring 2	1101100100110110
Mutated offspring 1	1100111000011110
Mutated offspring 2	1101101100110110

(a) (b) (c)

Figure 2. Examples of (**a**) crossover, (**b**) one-point crossover, (**c**) mutation operator [38].

There are different types of crossover namely: one-point crossover, two-point crossover, and multi-point crossover [39].

There are some evolving techniques used to accommodate some situations. A brief definition of the two types are as follows:

(a) One-point crossover: One-point crossover works when a crossover point along the chromosome is selected where genes are exchanged between the parents create two offspring (see Figure 2b).

(b) Two-point crossover: Here, two points are selected on the parent chromosomes. There is then the exchange of genes between the two points for the production of two offspring. The crossover operator is used to avoid the exact duplication of the parents from the old population in the new offspring. This ensures that the new population being produced through crossover operation is able to survive in the next generation and also has the desirable parts or qualities of the parents [40].

5. Mutation: Normally, mutation takes place after crossover is done. This operator applies the changes randomly to one or more “genes” to produce anew offspring, so it creates new adaptive solutions good avoid local optima. For example, in binary encoding, one or more randomly chosen bits can be switched from 0 to 1 or from 1 to 0; Figure 2c shows an example.

   Using the crossover operator alone to produce an offspring makes the GA stuck in the local optima, thus, the good parts of the parents survive in each generation, and the local optimal ones are to be found. This problem is called as the local-optima problem. The mutation operator is used to alleviate this problem by proving new offspring different from parents, and this encourages diversity in the population [41].
6. Termination (stopping) criteria: GA at the end must stop to announce the best solution in hand, there are several termination conditions that are used [42], those include:

   1. Fixed number of generations reached.
   2. The variance of the fitness evolves in the whole population is less than a predefined small quantity (fitness level has been reached [42].
   3. No improvement in the best fitness value [21].

2.1. GA Parameters

Determining the interactions that occur among different GA parameters has a direct impact on the quality of the solution, and keeping parameters values "balanced" improves the solution of the GA. There are four basic and important parameters used by GA, those include:

1. Crossover rate (probability): the number of times a crossover occurs for chromosomes in one generation, i.e., the chance that two chromosomes exchange some of their parts), 100% crossover rate means that all offspring are made by crossover. If it is 0%, then the complete new generation of individuals is to be exactly copied from the older population, except those resulted from the mutation process. Crossover rate is in the range of $[0, 1]$ [43].
2. Mutation rate (probability): this rate determines how many chromosomes should be mutated in one generation; mutation rate is in the range of $[0, 1]$. The purpose of mutation is to prevent the GA from converging to local optima, but if it occurs very often, GA is changed to random search [38,44].
3. Population size: the size of the population indicates the total number of the population’s inhabitants. Selection of population size is a sensitive issue; if the size of the population (search space) is small, this means little search space is available, and therefore it is possible to reach a local optimum. although, if the population size is very large, the area of search is increased and the computational load becomes high [45], therefore, the size of the population must be reasonable.
4. Number of generations: It refers to the number of cycles before the termination. In some cases, hundreds of loops are sufficient, but in other cases we might need more, this depends on the problem type and complexity. Depending on the design of the GA, sometimes this parameter is not used, particularly if the termination of the GA depends on specific criteria.

These GA parameters are determined whether the GA finds an optimal or near-optimal solution, and whether it finds an efficient solution [24]. Any change in the value of these parameters (increasing or decreasing) affects the result of GA negatively or positively [46,47]. Choosing the right parameters is a nontrivial task.

2.2. Crossover and Mutation Ratios

GA involves a process of complex interaction between its parameters, for example: using selection operators alone causes the GA to be stuck in copies of initial population without any new recreation of individuals. However, using selection and crossover operators converges the algorithm to better solutions, but does not guarantee optimality, more likely local optimal (sub-optimal) solutions. Therefore, mutation operator is needed to skip the local search space. Hence, parameters value in GA greatly determines the quality of the final solution [48]. Each operator of GA has a special and different influence. The impact of these factors is influenced by their probabilities, i.e., 100% probability for crossover gives totally different results from the 50%. The same applies to mutation probability. Therefore, the GA's design is limited to a choosing strategy for these parameters [38,48,49]. Keeping a balance between crossover and mutation rates is an issue of controlling parameters in GA. As a matter of fact, controlling values is one of the most important areas of research in GA that has been researched under a main topic known as (parameter setting) [50]. Parameter setting for GA is shown in Figure 3.

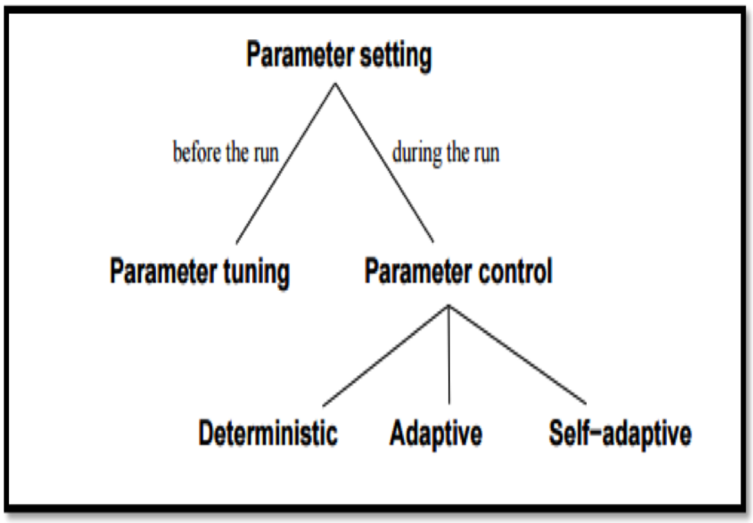


Figure 3. Global taxonomy of parameter setting [50].

There are two main types of setting parameter values, which are classified according to the parameter values behavior during the run: (i) The first one is parameter tuning (common approach), which depends on experimenting the different values for crossover and mutation, and then selecting the ones that give the best results before the final run of the algorithm. This operation is done without any change of the value during the run (fixed value) [50]. (ii) The second type is parameter control, which depends on initial parameter values for crossover and mutation that are changed in somehow in the run process. The techniques for changing the value of a parameter can be categorized to include the following:

1. Deterministic Parameter Control: used when the value of the parameter requires certain modifications using the same outcome rule. The parameter value is tuned to produce the same output without any results from the search process.
2. Adaptive Parameter Control: applied when there is a particular kind of feedback required from the search option that assists in altering the parameter.

3. Self-Adaptive Parameter Control. The GA here is allowed to develop its own parameters. The parameter values to be used are included in the individuals and go through mutation and crossover.

Those types of parameter setting are used by many researchers who attempt to find optimal or near-optimal solutions by using the best rates for crossover and mutation operators. This work fills in this gap, which is finding new strategies for controlling mutation and crossover rates. The proposed methods can be classified as deterministic parameter control.

3. Literature Review for Parameter Selection in GAs

Performance indicators to be used in rating of crossover and mutation are an important aspect in the designing of a GA. The key values of these parameters include; the appropriation of parameters and regulation of parameters. Therefore, over the years, many related works have been proposed to choose the best parameters for GA. In this section, a summary of the most important research conducted in the area is presented. In addition, we review some work related to the traveling salesman problem (TSP), as our proposed methods will be experimented to solve the TSP using GA.

3.1. Crossover and Mutation Rates with Population Size

Since the first mathematically formulated definition of GAs proposed by John Holland in 1975 in his book [1], many studies and pieces of research have been conducted in this field such as [51–53].

The crossover and mutation rates was proofed as key elements to success in the search in the GAs [51]. DeJong [54] suggested optimal range values for population size to be in the range of [50–100], mutation parameter rate to be (0.001), and high mutation rates leads the search to be random, the crossover used was based on one single point crossover to be around the rate of (0.6). Those parameters have been used in many GA implementations [54].

Grefenstette [32] designed a Meta-GA provided optimal parameters value for the simple GAs, and showed that small population size ranging from (20) to (40) is associated with high crossover rates went in harmony with low mutation rates. In general, his findings revealed that mutation rates above 0.05 were not useful for the optimal performance of (GAs). Accordingly, the researcher suggested another set of parameters with population size value of 30 individuals, a mutation rate of 0.01 and two-point crossover with 0.95 rates [32].

Schlierkamp-Voosen [55] investigated the dynamic behavior of mutation and crossover rates on GA regarding the binary functions. The GA parameter mutation and crossover rates were compared to their optimal solution performance. The mutation rates were most effective in small size populations. While the Crossover rates were depending on the size of the population, and mutation rates were the most robust search in small population size. Nevertheless, the (GA) combined the two operators in such a way that the performance was better than when using a single operator. The (DECEPTION) function showed that increasing the size of the population above to a certain number decreased the quality of the solutions obtained. The analysis of the interaction between mutation and crossover rates showed that the mutation rates normally increased with the size of population. Furthermore, it showed that the mutation rate = $1/n$ (where n was the size of problem) works efficiently. Using crossover and mutation together, achieved better results than using one of them alone. The crossover appeared to be more efficient than the mutation in larger population size, on the other hand, the mutation was more efficient in small populations [55].

Deb and Agrawal [26] investigated the performance of GA using several experiments that included simple to complex tests for the "onemax" problem. In their work, the researchers compared different (GAs) for several functional evaluations based on calculations and simulation results. The results of the test revealed that for solving simple problems such as "unimodal" or small modality problems, mutation operator played an important role despite crossover rates operator could also solve these problems. However, crossover and mutation rates (when applied alone) had two different effect zones for population size. The first test was conducted using the 0.9 crossover rates, the population size was in the range of (2–500), and a selection operator with (zero) mutation rate. The second test was conducted using the (zero) crossover rate with a population size in the range of (2–500), and with mutation rate of $[1, 0.5, 0.1]$. The last test combined the two parameters in various values where the mutation rates were $[1, 0.5, 0.1]$, the two values of crossover rates were $[0.9, 0.0]$. The conclusion of their experiment showed that the selection operators and mutation operators could perform well if accompanied with moderate population size. The selection operators and crossover operators without mutation performed well enough at larger population size than needed for GAs population with selection and mutation operators. In small population size, the researcher concluded that there is a need to use both crossover and mutation parameters as it is shown in Figure 4 [26].

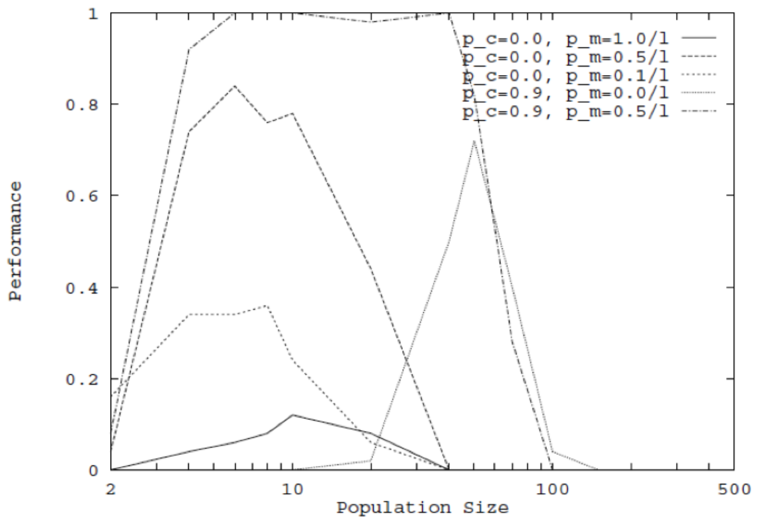


Figure 4. Performance of GAs using different GA parameter settings for “onemax” problem [26].

Hong et al. suggested an evolution algorithm called the Dynamic Genetic Algorithm (DGA) to automatically match the crossover and mutation rates according to each individual evaluation results in the new generation. More than one crossover and mutation rates are used simultaneously. The algorithm automatically chose the appropriate crossover and appropriate mutation types. The algorithm is a possible development of GAs performance although the GAs used a one-point crossover process as well a single process of mutation [56].

The population size and its impact on the genetic algorithm were extensively studied. Some researchers seem to agree that small population size could guide the (GA) to poor solutions [57–59]. Large population size necessitated that more computation search time was needed in finding an optimal or near optimal solution [59,60]. Rylander and Stanley [61] proposed a new methodology to determine the optimal population size in (GA). They did so by monitoring the difference in the results in each generation in different experiments. The experiments used two types of problems to test, namely the sort problem, and the maximum problem. They used (0.1) mutation rates, (0.5) crossover rates and several population sizes (100, 300, 400, 600). The results of the experiments showed the following: First, increasing the population size leads to an increase in the accuracy of the (GA), also the large population size provided a greater chance that the initial state population would contain an individual representing the optimal solution. Second, the large population size resulted in an increase in the number of generations’ convergence [61].

Diaz-Gomez, and Hougen [62] proposed a new approach to find a good initial state population for each small and large populations. The initial population encoded a possible solution to identify the problem. After creating the initial population individuals, each individual was evaluated by fitness value according to the fitness function for the sake of finding a good initial population. The method adopted dealt with small population for the sake of measuring diversity in a small portion of a large size population. They tried as well to generate chromosomes distributed randomly in the fitness landscape by suggesting a new idea using the center of mass, as an alternative method to measure diversity at the population level that depends on the population size, and chromosome length. This approach of analysis and measurement of the diversity of the individuals in the initial random population was important because of the relationship that stood between the initial population and other (GA) parameters and operators [62].

Dong and Wu [63] proposed a dynamic crossover and mutation rates. The crossover rates were calculated dynamically through calculating the Euclidean distance between two chromosomes [64]. They also calculated the Euclidean distance between the greatest and lowest fitness values of the chromosomes in the population. The process of crossover between individuals "long distance individuals" was effective, because of differences that appear among chromosomes, and this would avoid inbreeding and encourage diversity, and thus overcome convergence to local optima [63].

Chiroma et al. [65], conducted a survey of previous approaches and metrics related to the best values of the critical operators that control the quality of solutions. These operators were the mutation rates, crossover rates and the size of the population. They discussed the three operators in terms of decrease or increase in the proportion rates. They also discussed the relationship between mutation rates and crossover rates in the literature search. The small population size indicates insufficient sample sizes, which lead to poor solutions. Large population size indicates high efficiency in the search, and this was better than a small size of population, this also showed slow rate of convergence. The size of the population in this research was in the range of (12–4000). The low crossover rates lead to low rates of exploration. Choosing values between too high and very low crossover rates were to be used for effective implementation of (GA) ranged from (0.1–1). High mutation rates normally led to random search, and the mutation rates were always lower than the crossover rates, in which the mutation rate was in the range of (0.001–1) [65]. The following Table 1 contains GA parameter used to implement GA in previous literature.

Table 1. The GA parameters, PS - population size, CP - crossover probability, MP - mutation probability, used in previous works.

Reference	PS	CP	MP
[66]	30	0.95	0.01
[66]	80	0.45	0.01
[67]	100	0.9	1
[68]	100	0.8	0.005
[69]	50	0.9	0.03
[69]	50	1	0.03
[70]	50	0.8	0.01
[71]	15	0.7	0.05
[72]	100	1	0.003
[73]	30	0.8	0.07
[74]	100	0.8	0.01
[75]	500	0.8	0.2
[76]	40	0.6	0.1
[77]	100	0.8	0.3
[78]	100	0.6	0.02

Table 1. *Cont.*

Reference	PS	CP	MP
[79]	30	0.6	0.05
[80]	76	0.8	0.05
[67]	4000	0.5	0.001
[81]	30	0.6	0.001
[82]	50	0.8	0.2
[83]	50	0.9	0.1
[84]	30	0.9	0.1
[85]	20	0.8	0.02
[86]	50	0.9	0.01
[87]	20	0.9	0.3
[88]	330	0.5	0.5
[89]	100	0.8	0.1

3.2. Traveling Salesman Problem (TSP)

TSP is a high-end problem researcher in computer science [90–92]. It is simply easy to represent but hard to solve hence categorized under the NP-hard problems [93], which means no solution in polynomial time is discovered so far. A solution to the TSP purposes to obtain the closest path through a number of nodes that have the same start and endpoints in which all nodes are conferred and used once.

Many techniques can be used to solve the TSP, such as [94], (GA) [95–98], Tabu Search [99], Hill Climbing, Ant Colony [100], Particle Swarm [101], Simulated Annealing [102], Neural Networks [103], Elastic Nets [104], Nearest Neighbor and Minimum Spanning Tree algorithms [105], etc. Potvin [93] and Larrañaga et al. [95] conducted a survey of how to represent the traveling salesman problem (TSP), and also explained the advantages and disadvantages of different mutation and crossover rates and types. Singh and Singh [106] proposed an enhanced crossover operator for solving the traveling salesman problem (TSP).

In his study, Ahmed [107], proposed a method called sequential constructive crossover (SCX) to solve the TSP problem. The basic idea of this method was centered on choosing a point randomly called the crossover point. The method then was employed on sequential constructive crossover before the crossover point by using better edges. The rest of the genes after the crossover site were exchanged between parents to get two children; if there was already a gene, then, it was replaced with an unvisited gene [107].

TSP was considered to be a combination of optimized problems that was simple to describe but hard to solve, and it was categorized as the problems that could not be solved in polynomial time [90] which means it is attached to the NP-hard problems [93]. In 1832 an expert traveling salesman used the term "traveling salesman" in the manual for the traveling salesman that was published in Germany.

We discussed the previous studies that shed light on GA efficiency through developing different parameters of algorithm. Researchers stated that the key to evaluate the process was mainly through showing understanding of the interaction process between the GA parameters namely; crossover rates, mutation rates and population size. These parameters were related to each other in some way that affects the GA efficiency. Many studies agreed that diversity in the initial population with high crossover rates and low mutation rates was the best situation to apply GA. The goal of most of these studies was to bring diversity to the population using crossover and mutation rates to increase the efficiency of the GA.

4. Proposed Dynamic Approach

To improve parametric solutions for a complex system, there are two levels of the problem the system designer and the programmer need to worry about: first level of the problem in which a suitable algorithm is chosen. The second level of the problem, in which various parameters of the chosen algorithm should be chosen carefully to get the highest efficiency. If the chosen algorithm is non-parametric method, there is no need to worry about the second level.

Being a parametric method, GA needs to deal with the second level. When GAs is used in a wide variety of applications and complex systems, the second level task is needed to be applied to control its parameters for the sake of efficiency [32]. As mentioned above, the choice of mutation and crossover rates is well-known to be critical on the behavior and the performance of the (GA).

Many researchers use parameter tuning to set the parameter value in a way that necessitated doing the following: finding the good parameter value before the run (rigid parameters), then running the algorithm using these values. However, other researchers believed that the use of control parameter (Deterministic, adaptive) is more effective to give optimal or near optimal solutions. Some studies recommend using high crossover rates of about 80–95% with low mutation rates of about (0.1–1) [38,49].

In another study, the researcher recommends using a high rate of mutation (50%) with crossover together with small population [26].

Deterministic control parameter modifies the parameter value during the run without any feedback from the search process or user interaction. Accordingly, this type of parameter setting is to be used in an attempt to determine which behavior the parameter should deal with to be effective, and to try to find the optimal or the near optimal solution. Using high crossover rates with low mutation rates alongside large population size refers to the diversity in the large population; however, in the small population size, the mutation rates become lager in order to provide diversity and to increase the search quality [32], bringing diversity to the population using crossover rates and mutation rates to increase the efficiency of the GA.

Next, we propose two deterministic parameter control, followed by discussing two parameter tuning methods to be compared with the proposed methods later.

4.1. Dynamic Increasing of Low Mutation/Decreasing of High Crossover (ILM/DHC)

This deterministic method decreases the crossover ratios and increases the mutation ratios (ILM/DHC), where the crossover rate starts to decrease from 100% to 0% along the generations, at the same time it increases the mutation rate, which start from 0%, to be increased linearly to 100% along the process time.

When the algorithm starts to run Equations (1) and (2) change the mutation rate linearly, step by step according to the total number of generations, and the number of the current generation. To calculate mutation rates in ILM/DHC, the following equation is used:

$$MR = \frac{LG}{Gn} \qquad \text{where} \quad G = [1, 2, 3, \ldots, Gn], \tag{1}$$

$$M = MR * popsize, \tag{2}$$

where MR is the mutation rate, Gn is the total number of generations, LG the number of generation level (the current generation, when the ratio is calculated), M is the number of chromosomes that need to be mutated, "popsize" is population size. At the same time, Equations (3) and (4) change the crossover rate linearly, step by step according to the total number of generations, and the number of the current generation. To calculate crossover rates in (ILM/DHC), the following equation is used:

$$CR = 1 - \left(\frac{LG}{Gn}\right), \tag{3}$$

$$C = CR * popsize, \tag{4}$$

where CR is the crossover rate, and C is the number of chromosomes that need to be used for the crossover process. This method is used to provide different rates for both crossover and mutation in each generation level. Table 2 and Figure 5 show the rates during the run of GA.

Table 2. Crossover and mutation rates during the GA run using decreasing of high mutation (ILM) and increasing of low crossover (DHC).

Generation	**ILM/DHC**	
	IMR	**DCR**
1	0	1
100	0.0625	0.9375
200	0.125	0.875
300	0.1875	0.8125
400	0.25	0.75
500	0.3125	0.6875
600	0.375	0.625
700	0.4375	0.5625
800	0.5	0.5
900	0.5625	0.4375
1000	0.625	0.375
1100	0.6875	0.3125
1200	0.75	0.25
1300	0.8125	0.1875
1400	0.875	0.125
1500	0.9375	0.0625
1600	1	0

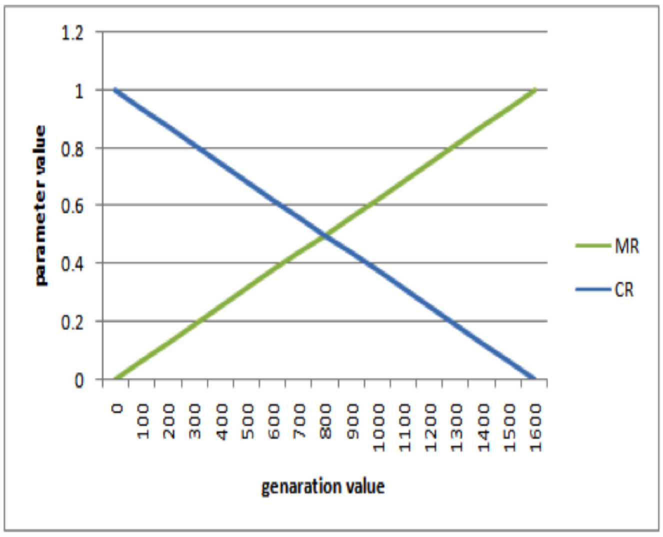


Figure 5. Crossover and mutation rates during the GA run using ILM/DHC approach.

This indicates that the crossover rate is decreased by a specific ratio in each generation level to become to 0% at the end of the algorithm in last generation. And Mutation rate is increased by a specific ratio in each generation level to reach 100% at the end of the GA run. As can be seen from Equations (1)–(4) the crossover rate is the complement of the mutation rate at each generation. Figure 5 depicts this relation, and the summation of crossover rate and mutation rate at the same generation level is equal to 100%. The goal of using dynamic changes in crossover and mutation rates is to bring diversity to the population during the run in a dynamic way.

Example 1. *The value of the mutation and crossover rates are calculated according to the generation level number. If the generation level is* 100*, the maximum generation* = 1600*, and population size is* = 100*, then the value of mutation and crossover rates according to Equations (1)–(4) are:*

$Generation\ Level = 100$, $Gn = 1600$, $MR = 100/1600 = 0.06$
$M = 0.06 * 100 = 6$ *individuals are mutated in generation level* 100*.*
$\#Generation = 100$, $Gn = 1600$, $CR = 1 - (100/1600) = 0.94$
$C = 0.94 * 100 = 94$ *individuals are used for the crossover process at generation level* 100*.*

4.2. Dynamic Decreasing of High Mutation Rate/Increasing of Low Crossover Rate (DHM/ILC)

This deterministic method is opposite to the ILM/DHC, where the mutation rate is decreased starting from 100%, and the crossover rate is increased starting from 0%. By using DHM/ILC when the GA runs Equations (5) and (7) change the rates linearly step by step according to generation number. As can be seen from Table 3 and Figure 6 both of the rates are complement of each other in the same step (generation), where the summation of crossover rate (CR), and mutation rate (MR) in the same generation level is equal to 100%.

Table 3. Crossover and mutation rates during the GA run using decreasing of high mutation (DHM) and increasing of low crossover (ILC).

Generation	DHM/ILC	
	DHM	**ILC**
1	1	0
100	0.9375	0.0625
200	0.875	0.125
300	0.8125	0.1875
400	0.75	0.25
500	0.6875	0.3125
600	0.625	0.375
700	0.5625	0.4375
800	0.5	0.5
900	0.4375	0.5625
1000	0.375	0.625
1100	0.3125	0.6875
1200	0.25	0.75
1300	0.1875	0.8125
1400	0.125	0.875
1500	0.0625	0.9375
1600	0	1

We can calculate the mutation rate in (DHM/ILC) using,

$$MR = 1 - (lG/Gn), \tag{5}$$

$$M = MR * popsize, \tag{6}$$

and calculate the crossover rate in (DHM/ILC) using,

$$CR = LG/Gn, \tag{7}$$

$$C = CR * popsize. \tag{8}$$

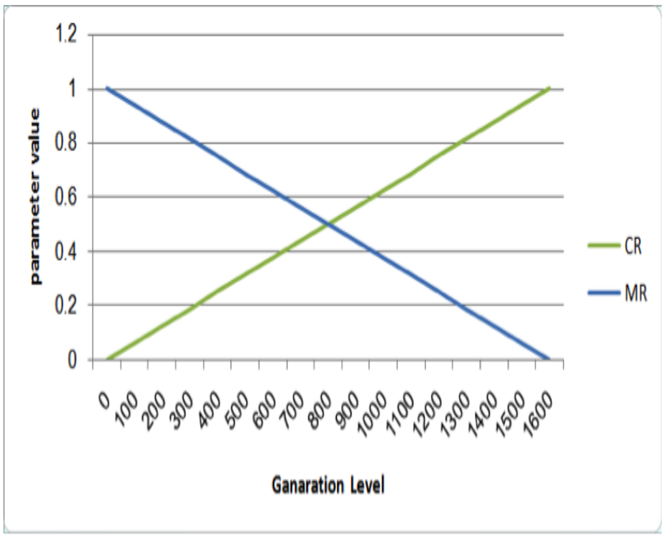


Figure 6. Crossover and mutation rates during the GA run using DHM/ILC approach.

Example 2. *The values of mutation and crossover rates are calculated according to the generation level number. If the generation level is* 500*, the maximum generation is* = 1600*, and the population size is* = 100*, then the value of mutation and crossover rates according to the previous equations are:*

$LG = 500$
$Gn = 1600$
$MR = 1 - (500/1600) = 0.69$
$M = 0.69 * 100 = 69$ *individuals are to be mutated in generation level* 500*.*
For Crossover rate: $LG = 500, Gn = 1600$, $CR = (500/1600) = 0.31$
$C = 0.31 * 100 = 31$ *individual are to be used for the crossover process at generation level* 500*.*

4.3. Fixed 50% for Mutation and Crossover Rates (FFMCR)

Starting from 0% to 100% and vice versa makes the average rate (along the run) of mutation and crossover using both DHM/ILC and ILM/DHC is 50% for each. This motivates us to use a tuning parameter method with the same average rate, thus having the same computation time, and a way to compare the proposed method with a tuning parameter method. These rates were used by some researchers, see for e.g., [88].

The FFMCR uses a fixed (50%) rate of both mutation and crossover rates. The parameter value in this approach is chosen before the run, and the algorithm continues in running using the same rate. This method shares the same time complexity with the proposed methods. The number of mutated individuals is constant for all generation levels. Also, the number of the crossovers is constant for all generation levels during the run as shown in Table 4 and Figure 7.

Table 4. Crossover and mutation rates during the GA run using FFMCR approach.

Generation	**Fifty-Fifty**	
	MR.5	**CR.5**
1	0.5	0.5
100	0.5	0.5
200	0.5	0.5
300	0.5	0.5
400	0.5	0.5
500	0.5	0.5
600	0.5	0.5
700	0.5	0.5
800	0.5	0.5
900	0.5	0.5
1000	0.5	0.5
1100	0.5	0.5
1200	0.5	0.5
1300	0.5	0.5
1400	0.5	0.5
1500	0.5	0.5
1600	0.5	0.5

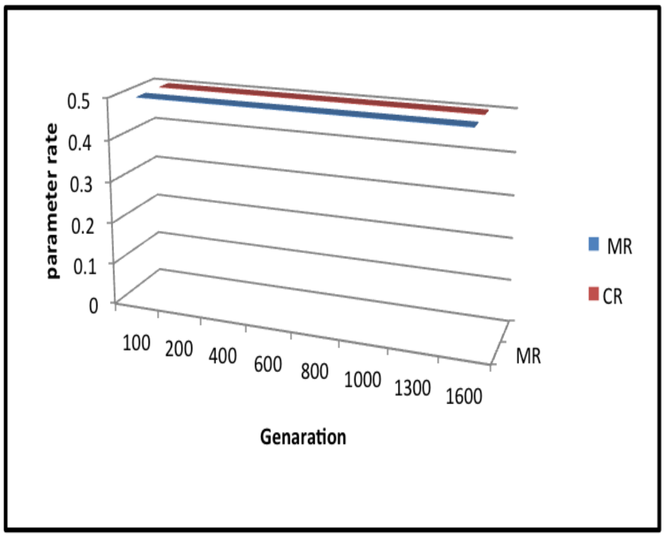


Figure 7. Crossover and mutation rates during the GA run using FFMCR approach.

4.4. Parameter Tuning Method (0.03 Mutation, 0.9 Crossover) (0.03MR0.9CR)

It is difficult (for the sake of this work) to use all the tuning parameters used in the literature; however, a fixed value for mutation rates is used where MR = 0.03, which is commonly used by some of GAs as explained in the literature review (Section 3), researchers used such rate include and not limited to: [69,72,77]. For the crossover rate we opt for CR = 0.9, because most of the GAs found in the literature used high rate for crossover, were most of the used CR in the range of 0.5 to 0.9. Researchers who used such rate include and not limited to: [69,83,84,86,87]. The number of mutated individuals is constant for all generation levels, also the number of individuals participated in crossover is constant for all generation levels during the run. We used the 0.03MR0.9CR as base for comparing our proposed work.

We proposed and discussed two new deterministic approaches for choosing mutation and crossover rates dynamically, namely DHM/ILC and ILM/DHC, in addition, two "parameter tuning" approaches to the same problem is presented, namely FFMCR and 0.03MR0.9CR, these were discussed to be compared with the proposed method in terms of solution goodness, and computational time consumed. Next, we provide detailed experimental results and discussion of our proposed dynamic approaches for the parameter selection in GAs.

5. Experimental Results and Discussion

Six sets of experiments were conducted to evaluate the proposed methods based on on the population size on different TSP problems; each set of experiments used four types of parameter settings, the proposed deterministic approaches; which are DHM/ILC and ILM/DHC. In addition, two parameter tuning approaches; namely FFMCR and 0.03MR 0.9CR. Different sizes of the population used to investigate the effect of the population size on the results using different parameter setting approaches. Population sizes used:

(1) Small population size (25 and 50); sets 1 and 2.
(2) Moderate population size (100 and 200); sets 3 and 4.
(3) Large population size (300 and 400); sets 5 and 6.

According to all the experiments, GA applied the reinsertion technique that is essentially sampling [63], where desirable qualities from both the new and the old generation individuals are chosen to be the population for the next generation. That is there is competition between the old and the new individuals.

The initial population seeding in all algorithms is random, and the selection strategy is a roulette wheel. The fitness function used by GA is a problem dependent; here we used the total distance of the TSP route, i.e., the summation of the distances between the sequence instances along the chromosome, the "best" solution is characterized by the minimum total distance, since the TSP is a minimization problem. The mutation operator used for all experiment sets is the exchange mutation [108]: which selects two genes randomly and switch between their positions. The crossover operator used is the Modified Crossover [109]. This crossover operator is an extension of the one-point crossover, which chooses the cut point at random from the first parent; then, the first child is created by appending genes (before cutting point) in the second parent to the first part. Finally, it repairs the rest of the gene to prevent duplicates. The termination criteria are based on a fixed number of generations reached, for all our experiments, the maximum number of generations was set to 1600. We used Path representation for encoding the TSP, where there is a natural representation for the tour [110], for example: a tour of city1 -> city3 -> city4 -> city2 -> city1 is represented in an integer array containing the cities' indexes: (1, 3, 4, 2).

The data set used for all sets of experiments includes 10 real TSP data; namely a280, u159, ch130, kroA100, pr76, berlin52, att48, eil51, pr144 and rat783, which were taken from the TSPLIB [111]. The number at the end of the name of each TSP indicates the number of instances (cities), each city is identified by its (x, y) location and a sequence number as shown in Figure 8. The chromosome, in this case, has random sequential numbers meaning that the chromosome length depends on the problem size *n* that is the size of each TSP problem. The result of the GA is the best solution that can be obtained, which is (in our case) the minimum total tour length, the word "result" and the "minimum total tour length" will be used interchangeably in this paper.

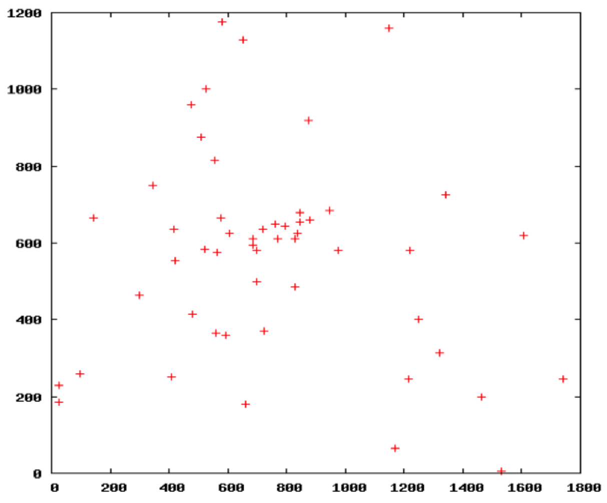


Figure 8. The Berlin52 problem in TSPLIB.

5.1. Set 1 of Experiments (Population Size = 25)

The aim of this experiment is to measure the performance of the proposed methods (DHM/ILC and ILM/DHC), compared to the tuning parameter approaches (0.03MR0.90CR and FFMCR) under a constant population size of 25 solutions (chromosome) for all the runs.

The GA is applied ten times, and the average of its "best" converged solution at each generation is recorded, starting from generation 100 up to generation 1600. Convergence to an optimal solution is beyond the purpose of this paper; however, we mean by convergence that the algorithm is providing better solutions while the generation number is increased.

Results from the first set of experiment showed that the best performance was recorded by the dynamic (ILM/DHC) rule, the "best performance" means here the ability to find the best solution on average, and the best solution is the minimum feasible rout for each TSP data. In most cases the performance of the dynamic (ILM/DHC) rule was better than the performance of the other methods, as is shown in Figures 9 and 10. The results of GA using different parameter setting methods on different TSP data are shown in Table 5 and Figure 11.

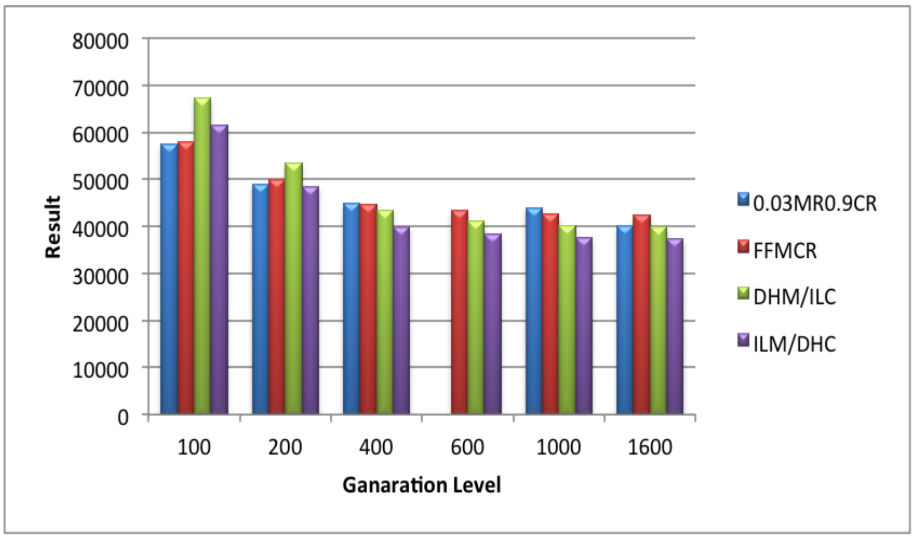


Figure 9. The average convergences of GA using different parameter setting approaches on TSP/att48, with population size = 25.

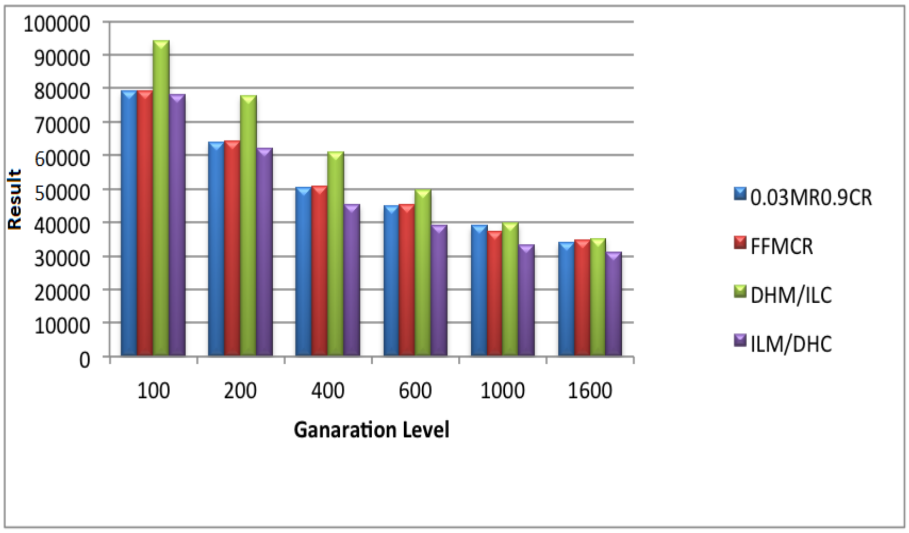


Figure 10. The average convergences of GA using different parameter setting approaches on TSP/kro100, with population size = 25.

Table 5. The average convergence of GA using approaches on all TSP data, after 1600 generations with population size = 25, the second number is the standard deviation.

Problem	0.03MR0.9CR	FFMCR	DHM/ILC	ILM/DHC
rat783	79,752/19,646	69,843/2280	69,124/1880	71,432/1295
pr144	20,0771/14,306	154,259/14,668	167,311/12,273	149,060/9876
eil51	501/35	497/29	492/24	465/10
berlin52	9606/569	9315/377	9068/491	8637/457
pr76	151,834/12,744	138,615/8629	146,682/14,217	138,071/8565
KroA100	34,126/2847	34,881/3701	35,237/2236	31,085/1564
att48	40,044/3113	42,511/3088	39,910/3261	37,507/1387
u159	105,188/6624	94,397/7485	92,660/7017	90,066/2806
a280	9181/379	8704/282	9177/440	9364/219
ch130	10,315/594	10,235/829	10,605/904	10,366/717

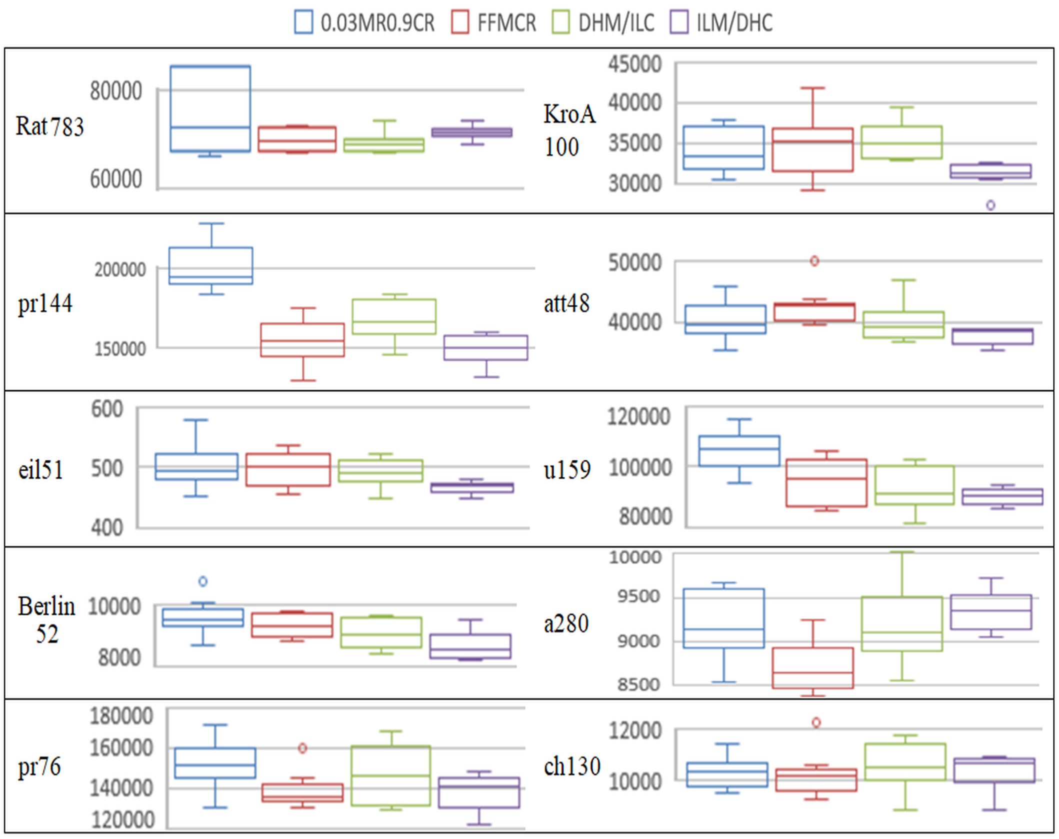


Figure 11. Box-and-whisker diagram of the results obtained using population size = 25.

As can be seen from Table 5 and Figure 11, the results indicate the efficiency of the Dynamic ILM/DHC over the other methods, where the Dynamic ILM/DHC performed the best in seven problems: pr76, berlin52, kroA100, att48, eil51, u159 and pr144, followed by the FFMCR, which performed the best in two problems: a280 and ch130. In this experiment, the population size is small (25), i.e., the diversity is not maintained we think that the Dynamic ILM/DHC provides diversity in small population, because starting the GA with a large crossover rate and small mutation rate with such a small population size, create similar offspring; however, while generation level goes up, the mutation rate become larger, and bring up different offspring, allowing the GA to skip local optima. this claim is also supported by the results from Figures 9 and 10, where the ILM/DHC started to take the lead significantly after generation 600 (Figures 9 and 10) that is acceptable, as a higher mutation rate is used, since in a small population higher mutation rate is needed to provide more diversity, to the small population, which had limited search space with just 25 individuals. The mutation started to be higher and more effective in generation (1000) to end the GA wining all the other methods.

5.2. Set 2 of Experiments (Population Size = 50)

The results from the second set of experiments showed that the best performance was also recorded by the Dynamic ILM/DHC rule, as we are still talking small population In most cases, the performance of the dynamic (ILM/DHC) rule was better than of the performance of the other method as is shown in Figures 12 and 13. The results of GA using different parameter setting methods on different TSP data with population size = 50 individuals are shown in Table 6.

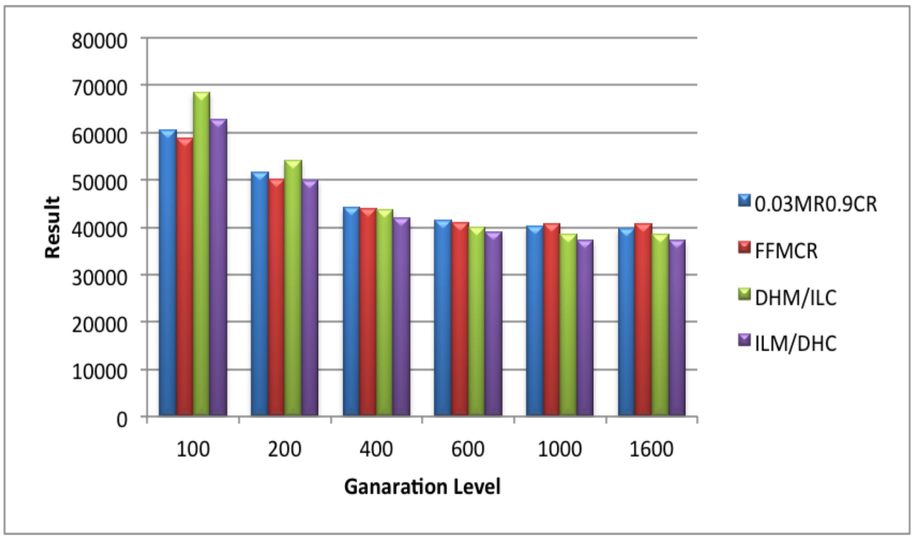


Figure 12. The average convergences of GA using different parameter setting approaches on TSP/att48, with population size = 50.

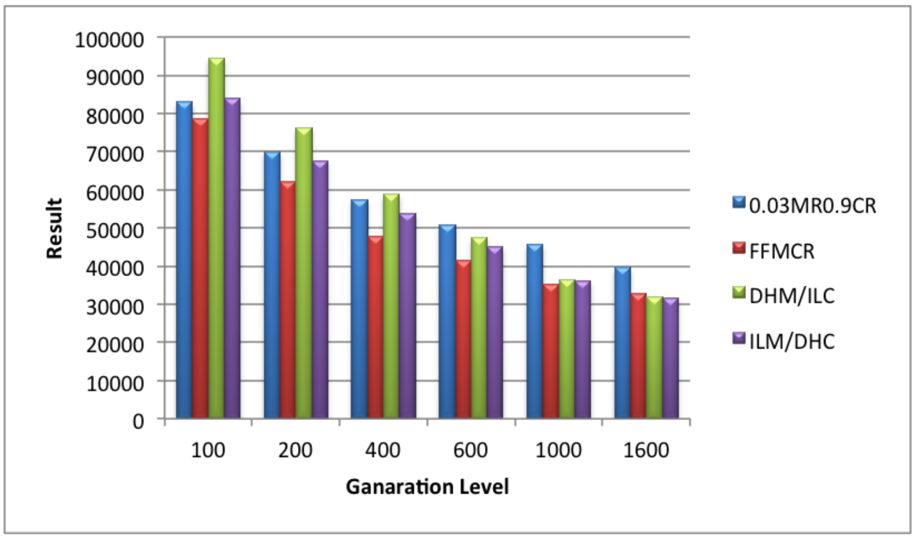


Figure 13. The average convergences of GA using different parameter setting approaches on TSP/KroA100, with population size = 50.

Table 6. The final average convergence of GA using different approaches on all TSP data, after 1600 generations with population size = 50.

Problem	0.03MR0.9CR	FFMCR	DHM/ILC	ILM/DHC
rat783	76,428	67,791.5	68,424.3	71,127.7
pr144	175,095.8	151,072.1	157,446.3	151,455.1
eil51	511.1	488.4	486	472.5
berlin52	9464	9155	9166.3	9024.5
pr76	155,182.4	140,766.9	139,644.6	135,316.6
KroA100	39,674.8	32,730.7	32,058.6	31,630.6
att48	39,872.2	40,686.3	38,448.1	37,244.2
u159	101,332.3	93,621.9	95,642.6	93,439.5
a280	9304.7	9264.5	9295.9	9067.3
ch130	11,455.4	10,231.1	10,383.2	10,178.5

As can be seen from Table 6, the results indicate the efficiency of the Dynamic ILM/DHC over the other methods where the Dynamic ILM/DHC provided the best solutions to: pr76, berlin52, kroA100, att48, eil51, u159, a280 and ch130 followed by FFMCR that performed the best on: rat783, pr144.

In this set of experiments, the population size is still small (50). Therefore, we stick with the aforementioned explanation of the efficiency of the proposed ILM/DHC when it is applied on small populations. Results from Figures 12 and 13 also supports the same explanation, as ILM/DHC approach started to take the lead after (1000) generations, i.e., a higher mutation ratio to support diversity.

5.3. Set 3 of Experiments (Population Size = 100)

The results from the third set of experiments show that the best performance result was recorded by the dynamic (ILM/DHC) approach. In most cases, the performance of the dynamic (ILM/DHC) rule was better than of the performance of the other method as is shown in Figures 14 and 15. The results of GA using different parameter setting methods on different TSP data with population size = 100 individuals are shown in Table 7.

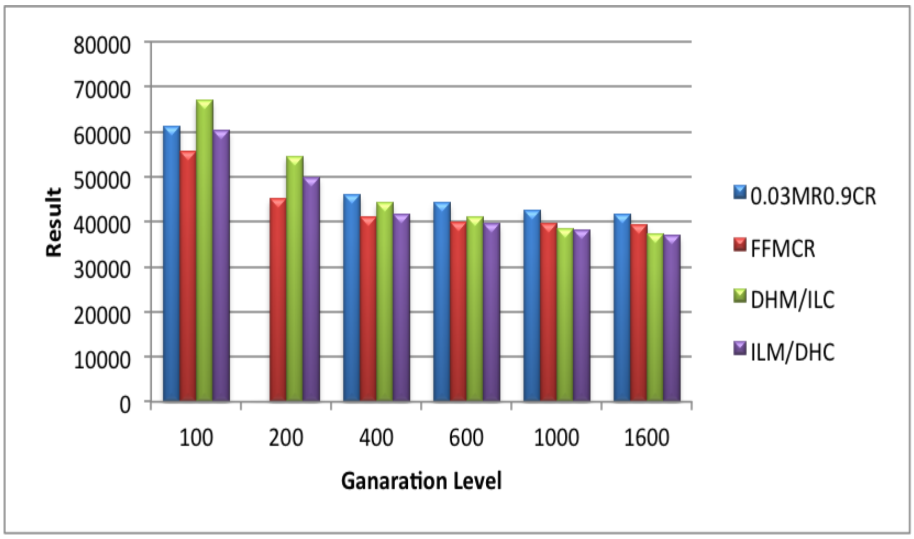


Figure 14. The average convergences of GA using different parameter setting approaches on TSP/att48, with population size = 100.

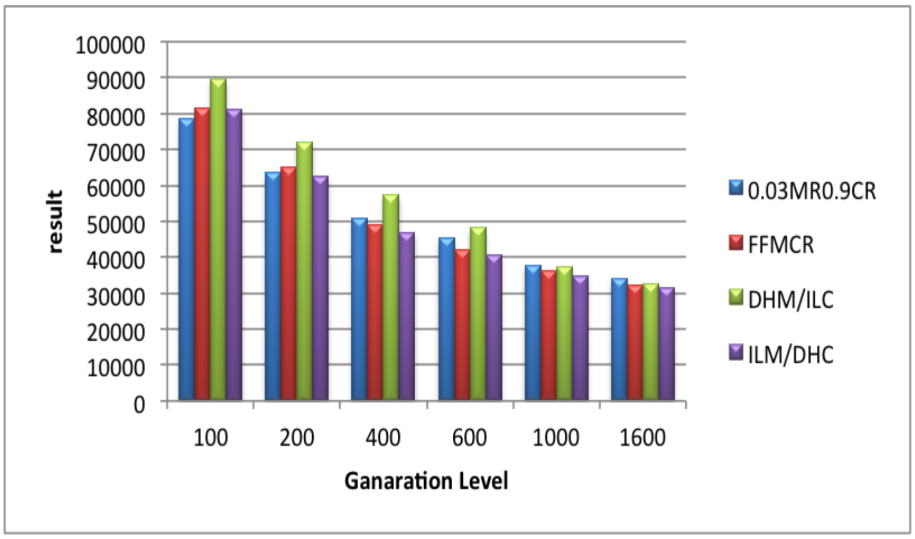


Figure 15. The average convergences of GA using different parameter setting approaches on TSP/KroA100, with population size = 100.

Table 7. The final average convergence of GA using different approaches on all TSP data, after 1600 generations with population size = 100.

Problem	0.03MR0.9CR	FFMCR	DHM/ILC	ILM/DHC
rat783	70,183.6	68,322.4	67,795.7	71,098.2
pr144	172,820.9	158,910.4	156,527.1	153,154
eil51	496.8	481.1	479	483.1
berlin52	9200.3	8667.2	8782.5	8820.1
pr76	148,600.5	146,552.8	149,661.5	139,733.7
KroA100	34,191	32,176.1	32,549.2	31,598.7
att48	41,811.1	39,459.3	37,355.8	36,929.5
u159	98,011.6	90,671.5	95,826.5	98,659
a280	8957.1	8897	9134.1	9457.7
ch130	11,480	10,609.8	10,253.3	10,192.4

As can be seen from Table 7, the results still indicate the efficiency of the Dynamic ILM/DHC over the other methods where the Dynamic ILM/DHC provided the best solutions to: pr76, kroA100, att48, sh130, and pr144, followed by FFMCR berlin52, u159 and a280, followed by DHM/ILC, which started to perform better compared to smaller populations In this experiment, the population size is moderate (100), and therefore, the ILM/DHC did not perform better than it used to do in smaller populations, in addition, the difference in the quality of the provided solutions from the other methods is not significant (see Figures 14 and 15), and only won 5 TSPs, this means that the ILM/DHC started to lose some of its reputation as we increase the size of the population, since diversity is partially maintained in moderate population sizes. According to the results of the previous three sets of experiments, the Dynamic ILM/DHC recorded the best results in small sized populations (25, 50), nevertheless it start to become weaker in moderate sized populations (100). This triggers the following question: What if the population size got larger?

The answer of this question we need further sets experiments using larger population sizes.

5.4. Set 4 of Experiments (Population Size = 200)

The results of the fourth set of experiments show that the best performance was recorded by the Dynamic DHM/ILC approach. In most cases, the Dynamic DHM/ILC outperformed the other approaches as shown in Figures 16 and 17. The results of GA using different parameter setting methods on different TSP data with population size = 200 individuals are shown in Table 8.

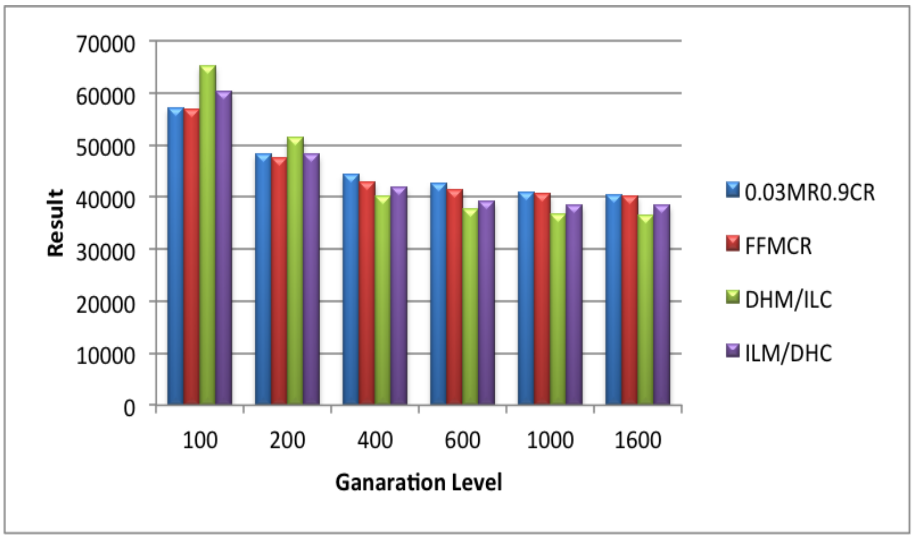


Figure 16. The average convergences of GA using different parameter setting approaches on TSP/att48, with population size = 200.

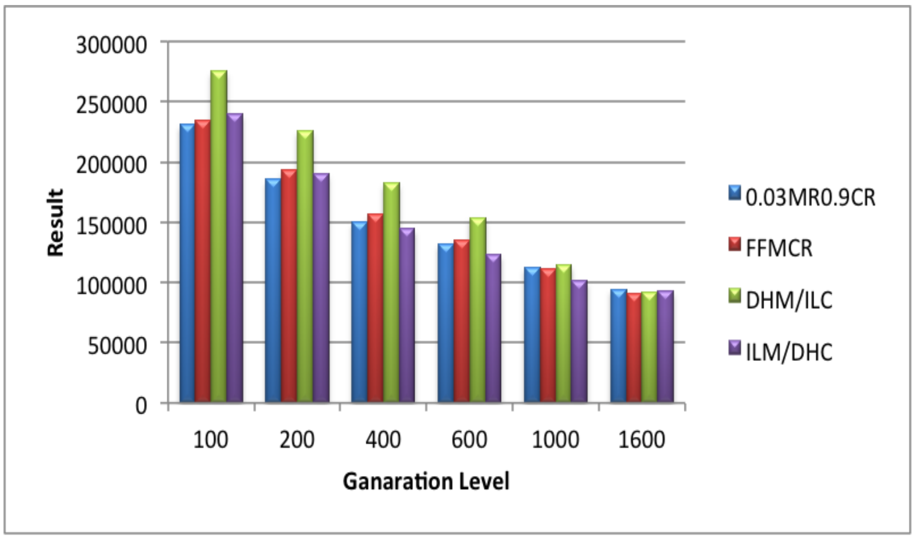


Figure 17. The average convergences of GA using different parameter setting approaches on TSP/u159, with population size = 200.

Table 8. The average convergence of GA using different approaches on all TSP data, after 1600 generations with population size = 200.

Problem	**0.03MR0.9CR**	**FFMCR**	**DHM/ILC**	**ILM/DHC**
rat783	70,486.3	71,100.3	67,399.3	71,571.3
pr144	169,101.1	177,072.5	155,994.3	165,080.7
eil51	501.5	490.7	468.5	481.6
berlin52	9697.6	9397.8	9052.3	9526.9
pr76	151,172.8	149,717.8	139,640.5	142,065.1
KroA100	34,587.6	35,331.6	32,754.5	32,435.6
att48	40,504.6	40,272	36,502.1	38,514.7
u159	94,578.1	91,125.4	91,861.1	92,766.4
a280	9504.4	8912.6	9175.8	9425.4
ch130	10,442.9	10,238.1	10,687.2	10,213.4

As can be seen from Table 8, the results indicate the efficiency of the Dynamic DHM/ILC over the other methods where the Dynamic DHM/ILC performed the best solution to the problems: pr144, pr76, eil5, berlin52, att48, followed by the FFMCR and the ILM/DHC that performed the best in two problems in addition. A population with 200 individuals is a large population. In addition, therefore the diversity is almost maintained, and GA does not need high mutation rate to increase diversity, on the contrary it needs more crossovers to generate better offspring from the diverse large sized population, particularly, when the GA process goes to higher level of generations. This result is expected, since we noticed the degradation in the performance of the ILM/DHC as the population size goes larger, knowing that the ILM/DHC approach is opposite to the DHM/ILC approach.

5.5. Set 5 of Experiments (Population Size = 300)

The results from this set of experiments show that the best performance is also recorded by the Dynamic DHM/ILC approach. In most cases the performance of the Dynamic DHM/ILC was better than of that of the other approaches, this is shown in Figures 18 and 19. The results of GA using different parameter setting methods on different TSP data with population size = 300 individuals are shown in Table 9.

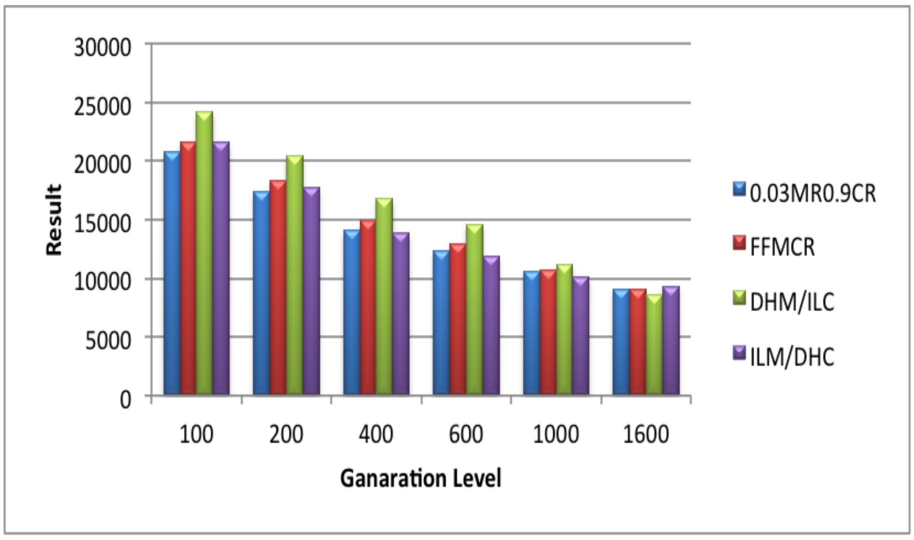


Figure 18. The average convergences of GA using different parameter setting approaches on TSP/a280, with population size = 300.

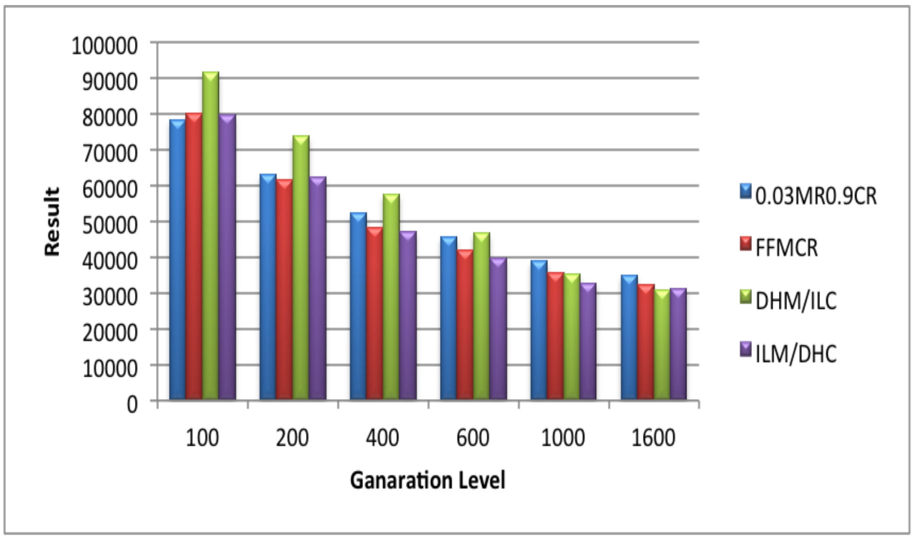


Figure 19. The average convergences of GA using different parameter setting approaches on TSP/kroA100, with population size = 300.

Table 9. The average convergence of GA using different approaches on all TSP data, after 1600 generations with population size = 300.

Problem	0.03MR0.9CR	FFMCR	DHM/ILC	ILM/DHC
rat783	69,554.8	70,979.8	68,091.9	72,438.5
pr144	178,213.8	149,154.5	148,610.7	157,487.6
eil51	521.9	496.3	475.1	494.2
berlin52	9642.6	9136.7	8790.7	9207
pr76	159,528.7	146,208.2	133,732.8	140,053.7
KroA100	35,143	32,257.4	31,003.1	31,395
att48	40,653.1	40,671.7	38628.1	38,613.4
u159	95,409.8	94,542.2	91,893.9	96,699.1
a280	9017.8	9031.9	8645.6	9352.6
ch130	13,084.6	10,364.4	9912	10,687

As can be seen from Table 9, the results indicate the efficiency of the Dynamic DHM/ILC over the other approaches where the Dynamic DHM/ILC performed the best solutions for the problems:

rat783, pr76, eil51, berlin52, KroA100, ch130, u159, a280 and pr144 followed by the Dynamic ILM/DHC in problem: att48. This show the domination of the DHM/ILC in larger populations and support the previous set of experiments, giving more stable results as increasing the size of the population, the aforementioned explanation still applies for the same mentioned reasons.

5.6. Set 6 of Experiments (Population Size = 400)

The Results from this experiment shown that the best performance is still recorded by the Dynamic DHM/ILC approach. In all the tested cases, the performance of the Dynamic DHM/ILC was better than those of other approaches, the performance of all approaches is shown in Figures 20 and 21. results are averaged over 10 runs. The results of GA using different parameter setting methods on different TSP data with population size = 300 individuals are shown in Table 10.

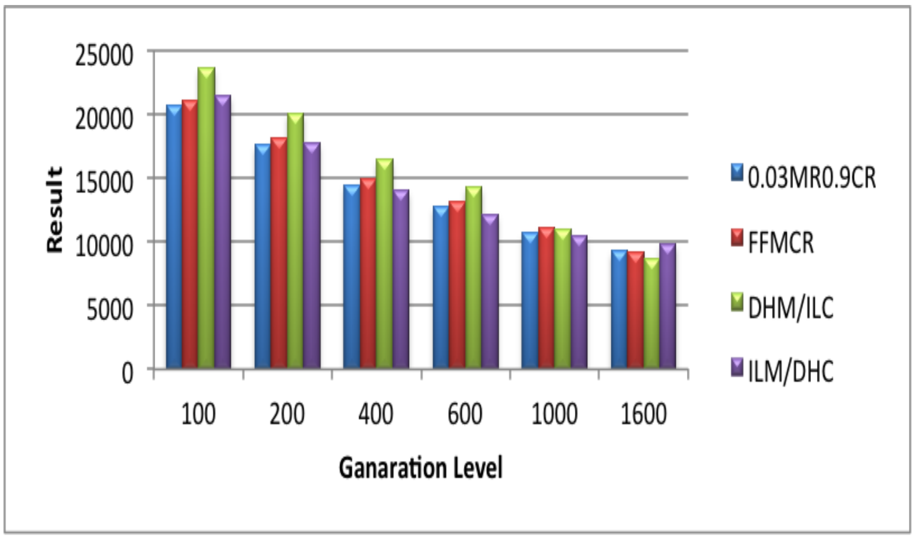


Figure 20. The average convergences of GA using different parameter setting approaches on TSP/a280, with population size = 400.

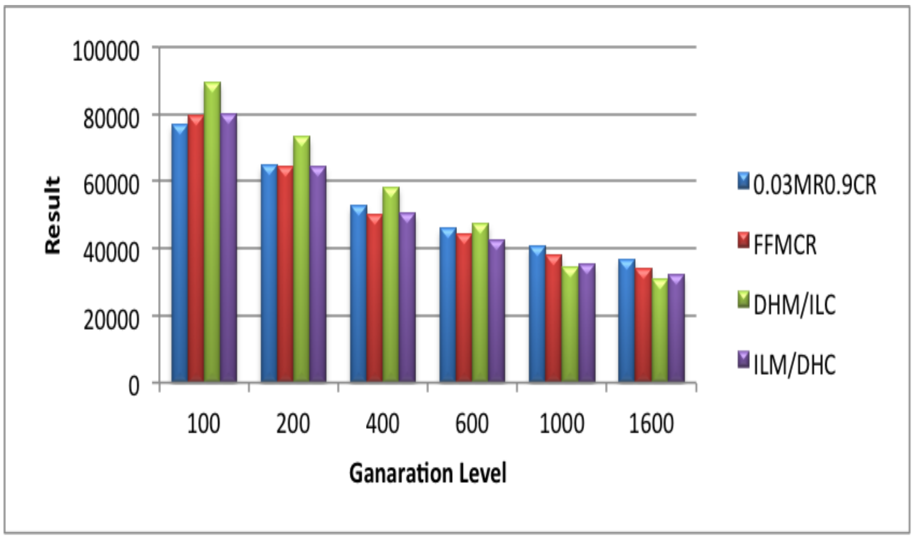


Figure 21. The average convergences of GA using different parameter setting approaches on TSP/kro100 with population size = 400.

Table 10. The average convergence of GA using different approaches on all TSP data, after 1600 generations with population size = 400.

Problem	0.03MR0.9CR	FFMCR	DHM/ILC	ILM/DHC
rat783	71,517.6	69,253.6	67,424.3	73,022.8
pr144	174,357	149,038.5	146,385.7	158,169.1
eil51	519.6	498.7	476.6	485.4
berlin52	9250.7	9167.6	8619.7	9166
pr76	152,704.2	142,986.3	135,876.6	141,485.1
KroA100	36,751.3	34,094.3	30,804.1	32,291.9
att48	43,077.1	41,121.3	36,854.5	39,310.4
u159	101,313.2	94,858.3	87,507.6	96,255.1
a280	9287.7	9177.3	8698.4	9862.1
ch130	11,526.7	10,247.1	9854.9	10,622

As can be seen from Table 10 and Figure 22, the results indicate the superiority of the Dynamic DHM/ILC over all the other approaches, where the Dynamic DHM/ILC provided the best solution for all problems used in the experiments. Moreover, the provided solutions are significant compared to the other approaches, i.e., the difference between the best solutions (provided by the DHM/ILC) and the other solutions (provided by other approaches) is high and significant. Same explanation provided in the previous sets of experiments applies here, the only difference is that the population size has become larger (400), and therefore the diversity is guaranteed, so the need for more crossovers at the end of the GA is vital, and much important that the number of mutations, which yields diverse but weak solutions, while the crossover yields stronger solutions as larger parts of the "good" chromosomes are exchanged to provide even better individuals. Another thing that worth mentioning is the result of the most common approach for GA parameter setting, which is the 0.03MR0.9CR, this approach was used extensively by researchers who used GA for solving optimization problems, we found with this number of experiment that this approach never won any problem except one problem (rat783) in experiment set 4 with population size 200.

More results from the six sets of experiments can be seen at Appendix A.

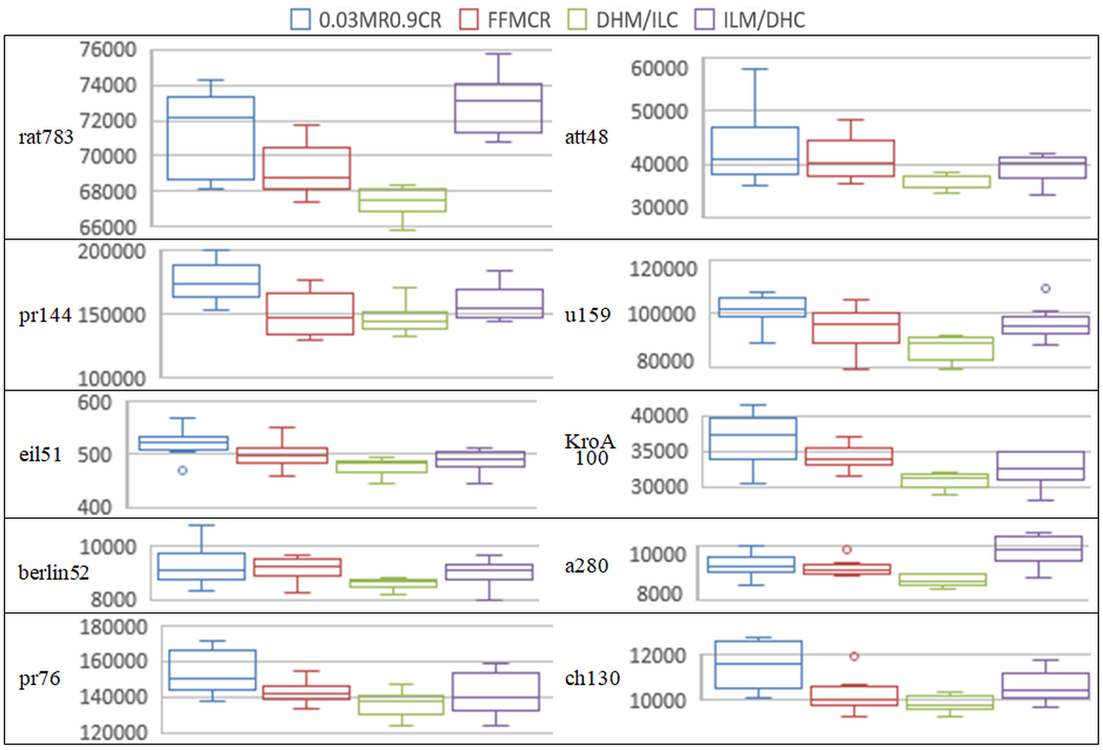


Figure 22. Box-and-whisker diagram of the results obtained using population size = 400.

5.7. Time Complexity

Time complexity of the proposed work in comparison with other methods. The time complexity of the (GAs) depended mainly on the representation of the chromosomes (the size of the chromosome, which varied depending on the problem), the genetic parameters (number of generations, population size, mutation probability and crossover probability), and the selection time [30]. Consequently, the time complexity of the (GA) is calculated as follows: Suppose that the size of population is (P), the number of generations is (G), the probability of crossover and mutation were (PC and PM) respectively, and the chromosome size is (n), which is the variable number of instances (TSP cities in our case), normally,

$$n >> a\ constant.$$

The Quick Sorting algorithm is used to sort out the population and to take the best solutions. The number of operations needed for the selection process was:

$$(P \times \log(P))$$

The number of the mutated chromosomes was equal to

$$(PM \times P).$$

Since we need to go along each chromosome; the number of operations needed for the mutation process is:

$$(PM \times P \times n)$$

The number of crossovers is equal to

$$(PC \times P)$$

However, Crossover operator is a binary variation operation, i.e., it works on 2 chromosomes, and yield two chromosomes (offspring), while the mutation operator is a unary variation operator, works on one chromosome and yield only one chromosome, therefore The number of crossovers is equal to

$$(2 \times PC \times P).$$

Since we need to go along each chromosome; the number of operations needed for the crossover process is:

$$(2 \times PC \times P \times n).$$

All the previous operations were repeated (G) times. Therefore, the time complexity of the overall (GA) is:

$$\begin{aligned} T = G((P \times \log(P)) + (PM \times P \times n) \\ +(2 \times PC \times P \times n)) \end{aligned} \tag{9}$$

Since, G and P were constant (ks) regardless of what the problem was, assuming that $k1 = G$, $k2 = P$, $k3 = PM \times P$, $k4 = PC \times P$ then theoretically: k = k1 = k2 = k3 = k4, and therefore (GA)TC became:

$$T = k((k \times \log(k)) + (k \times n) + (k \times n)) \tag{10}$$

$$= k^2 \log(k) + 2nk^2 \tag{11}$$

Since k was a constant, then k^2 and $\log(k)$ were also constants, which makes the Equation (10) as the following:

$$T = k + (kn) \tag{12}$$

Therefore, the time complexity of the traditional (GAs) is linear (of order (n)), i.e., T depended mainly on the problem size (chromosome size) as it is the only variable in the process:

$$T = O(n) \tag{13}$$

However, in practice, we feel that some approaches are slower than others, so we need to analyze the time complexity of the four approaches depending on Equation (9). Assume that time complexity for DHM/ILC is T1, and time complexity for ILM/DHC is $T2$. The average of crossover rates used by DHM/ILC is

$$\begin{aligned} AV &= (0\% + 1\% + 2\% + \ldots + 100\%)/100 \\ &= (100(100+1)/2 * 100)/100 \end{aligned}$$

According to the Gaussian closed form

$$AV = (100(100+1)/(2*100)/100 = 50\%$$

And the average of the mutation rate is the same; hence they are complements of each other, and

$$T1 = T2$$

Because they have the same ratios but opposite to each other Recalling Equation (9) we get,

$$\begin{aligned} T1 &= T2 \\ &= G((P \times \log(P)) + (50\% \times P \times n) \\ &+ (2 \times 50\% \times P \times n)) \\ &= G \times ((P\log(P)) + (1.5P \times n)) \end{aligned} \tag{14}$$

Assuming that T3 represents time consumed by FFMCR, since it uses 50

$$T3 = T1$$

Assuming that T4 represents the time consumed by 0.03MR0.9CR, recalling Equation (9) we get,

$$\begin{aligned} T4 &= G((P \times \log(P)) + (3\% \times P \times n) \\ &+ (2 \times 90\% \times P \times n)) \\ &= G \times ((P\log(P)) + (1.74P \times n)) \end{aligned} \tag{15}$$

As can be seen from Equations (14) and (15), the number of operations needed by the proposed method is less than those of the 0.03MR0.9CR, and therefore the proposed approaches for choosing the mutation and crossover ratios is faster than the most common predefined rates (0.03MR0.9CR).

6. Conclusions

We reviewed GA representations and parameter selection in GAs along with new deterministic approaches were proposed to change the crossover and mutation rates parameters dynamically. Without any feedback or user interaction, the change in parameters value linearly based on mathematical equations, namely ILM/DHC and DHM/ILC. Based on several sets of experiments on real data from the TSPLIB, the Dynamic ILM/DHC was the best to operate with small population size, attaining the best solutions if compared with FFMCR, 0.03MR0.9 CR and DHM/ILC in small population (25, 50, 100). The success of this algorithm in small sized population is due to lack of diversity of such small search space, and therefore, by the increasing level of generation, and the extensive use of crossover operations the final solutions become almost similar, and in need for a large number of mutations, which is provided to the GA just on time, preventing the algorithm from stuck at a local optima, and therefore increases the performance of the GA with small sized population. The results also showed that the Dynamic DHM/ILC performed very well with large population sizes (200, 300, 400) if compared with other methods. this is due to the need for more crossovers at the end of the algorithm at higher levels of generation, as the large sized populations are diverse by nature, and therefore do not need many mutations to get the algorithm from a local optima, on the contrary it needs more stronger solutions that depends on exchanging larger parts from good parents to provide better offspring, and this enhance the final solution of the GA with large populations.

We note also that the 0.03MR0.9CR approach (which is one of the most used predefined parameter setting) did not perform the best in any set of experiments, nor with any population, so we think that using such predefined rates for both crossover and mutation is not a good practice, in terms of quality of solutions and time complexity. In the future, we intend to extend the proposed approaches to be used for GA parameter setting for other optimization problems, such as the maxone, the knapsack problem, scheduling problems, etc.

The experimental results show averages across 10 runs, and we draw conclusions from small differences in these averages. If the scatter is too strong, these conclusions might not be well-founded, therefore, we plan to increase the number of runs to be at least 30 runs for each experiment in the future. Moreover, the linearity of the approaches needs to be further investigated, and compared with other adaptive and self-adaptive approaches, in addition to use indexing techniques such as [112–115].

Author Contributions: Conceptualization, A.H. (Ahmad Hassanat) and E.A. (Esra'a Alkafaween); methodology, A.H. (Ahmad Hassanat) and K.A. (Khalid Almohammadi); software, A.H. (Ahmad Hassanat); validation, E.A. (Eman Abunawas), E.A. (Esra'a Alkafaween) and K.A. (Khalid Almohammadi); formal analysis, A.H. (Awni Hammouri) and V.B.S.P.; investigation, E.A. (Eman Abunawas); resources, E.A. (Eman Abunawas); data curation, E.A. (Esra'a Alkafaween); writing—original draft preparation, A.H. (Ahmad Hassanat) and V.B.S.P.; writing—review and editing, K.A., A.H. (Ahmad Hassanat) and V.B.S.P.; visualization, E.A. (Eman Abunawas); supervision, A.H. (Ahmad Hassanat); project administration, A.H. (Ahmad Hassanat).

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. Results from the 6 Sets of Experiments Conducted for this Study

More results from the 6 sets of experiments conducted for this study are given below in Tables A1–A10.

Table A1. The solution average in TSP (berlin52) problem, four ratios with several population sizes.

Pop Size	Generation	Dynamic DHM/ILC	Dynamic ILM/DHC	0.03MR0.9CR	FFMCR
400	1600	8619.7	9166	9250.7	9167.6
300	1600	8790.7	9207	9642.6	9136.7
200	1600	9052.3	9526.9	9697.6	9397.8
100	1600	8782.5	8820.1	9200.3	8667.2
50	1600	9166.3	9024.5	9464	9155
25	1600	9068.2	8637.4	9606.1	9314.8

Table A2. The solution average in TSP (pr76) problem, four ratios with several population sizes.

Pop Size	Generation	Dynamic DHM/ILC	Dynamic ILM/DHC	0.03MR0.9CR	FFMCR
400	1600	135,876.6	141,485.1	152,704.2	142,986.3
300	1600	133,732.8	140,053.7	159,528.7	146,208.2
200	1600	139,640.5	142,065.1	151,172.8	149,717.8
100	1600	149,661.5	139,733.7	148,600.5	146,552.8
50	1600	139,644.6	135,316.6	155,182.4	140,766.9
25	1600	146,682.1	138,070.6	151,834.4	138,614.6

Table A3. The solution average in TSP (KroA100) problem, four ratios with several population sizes.

Pop Size	Generation	Dynamic DHM/ILC	Dynamic ILM/DHC	0.03MR0.9CR	FFMCR
400	1600	30,804.1	32,291.9	36,751.3	34,094.3
300	1600	31,003.1	31,395	35,143	32,257.4
200	1600	32,754.5	32,435.6	34,587.6	35,331.6
100	1600	32,549.2	31,598.7	34,191	32,176.1
50	1600	32,058.6	31,630.6	39,674.8	32,730.7
25	1600	35,236.8	31,084.9	34,126.1	34,881.2

Table A4. The solution average in TSP (u159) problem, four ratios with several population sizes.

Pop Size	Generation	Dynamic DHM/ILC	Dynamic ILM/DHC	0.03MR0.9CR	FFMCR
400	1600	87,507.6	96,255.1	101,313.2	94,858.3
300	1600	91,893.9	96,699.1	95,409.8	94,542.2
200	1600	91,861.1	92,766.4	94,578.1	91,125.4
100	1600	95,826.5	98,659	98,011.6	90,671.5
50	1600	95,642.6	93,439.5	101,332.3	93,621.9
25	1600	92,659.6	90,066.3	105,187.9	94,397.3

Table A5. The solution average in TSP (ch130) problem, four ratios with several population sizes.

Pop Size	Generation	Dynamic DHM/ILC	Dynamic ILM/DHC	0.03MR0.9CR	FFMCR
400	1600	9854.9	10,622	11,526.7	10,247.1
300	1600	9912	10,687	13,084.6	10,364.4
200	1600	10,687.2	10,213.4	10,442.9	10,238.1
100	1600	10,253.3	10,192.4	11,480	10,609.8
50	1600	10,383.2	10,178.5	11,455.4	10,231.1
25	1600	10,605	10,366.3	10,315.1	10,235.4

Table A6. The solution average in TSP (a280) problem, four ratios with several population sizes.

Pop Size	Generation	Dynamic DHM/ILC	Dynamic ILM/DHC	0.03MR0.9CR	FFMCR
400	1600	8698.4	9862.1	9287.7	9177.3
300	1600	8645.6	9352.6	9017.8	9031.9
200	1600	9175.8	9425.4	9504.4	8912.6
100	1600	9134.1	9457.7	8957.1	8897
50	1600	9295.9	9067.3	9304.7	9264.5
25	1600	9176.5	9364.1	9180.7	8703.6

Table A7. The solution average in TSP (att48) problem, four ratios with several population sizes.

Pop Size	Generation	Dynamic DHM/ILC	Dynamic ILM/DHC	0.03MR0.9CR	FFMCR
400	1600	36,854.5	39,310.4	43,077.1	41,121.3
300	1600	38,628.1	38,613.4	40,653.1	40,671.7
200	1600	36,502.1	38,514.7	40,504.6	40,272
100	1600	37,355.8	36,929.5	41,811.1	39,459.3
50	1600	38,448.1	37,244.2	39,872.2	40,686.3
25	1600	39,909.6	37,507.3	40,044.2	42,510.5

Table A8. The solution average in TSP (eil51) problem, four ratios with several population sizes.

Pop Size	Generation	Dynamic DHM/ILC	Dynamic ILM/DHC	0.03MR0.9CR	FFMCR
400	1600	476.6	485.4	519.6	498.7
300	1600	475.1	494.2	521.9	496.3
200	1600	468.5	481.6	501.5	490.7
100	1600	479	483.1	496.8	481.1
50	1600	486	472.5	511.1	488.4
25	1600	491.7	465.4	501	496.8

Table A9. The solution average in TSP (pr144) problem, four ratios with several population sizes.

Pop Size	Generation	Dynamic DHM/ILC	Dynamic ILM/DHC	0.03MR0.9CR	FFMCR
400	1600	146,385.7	158,169.1	174,357.6	149,038.5
300	1600	148,610.7	157,487.6	178,213.8	149,154.5
200	1600	155,994.3	165,080.7	169,101.1	177,072.5
100	1600	156,527.1	153,154.0	172,820.9	158,910.4
50	1600	157,446.3	151,455.1	175,095.8	151,072.1
25	1600	167,311.4	146,634.9	200,770.7	154,258.6

Table A10. The solution average in TSP (rat783) problem, four ratios with several population sizes.

Pop Size	Generation	Dynamic DHM/ILC	Dynamic ILM/DHC	0.03MR0.9CR	FFMCR
400	1600	67,424.3	73,022.8	71,517.6	69,253.6
300	1600	68,091.9	72,438.5	69,554.8	70,979.8
200	1600	67,399.3	71,571.3	70,486.3	71,100.3
100	1600	67,795.7	71,098.2	70,183.6	68,322.4
50	1600	68,424.3	71,127.7	76,428	67,791.5
25	1600	69,123.6	71,432.3	79,752	69,843.1

References

1. Holland, J.H. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*; MIT Press: Cambridge, MA, USA, 1975.
2. Man, K.F.; Tang, K.S.; Kwong, S. Genetic algorithms: Concepts and applications. *IEEE Trans. Ind. Electron.* **1996**, *43*, 519–534. [CrossRef]
3. Golberg, D.E. *Genetic Algorithms in Search, Optimization, and Machine Learning*; Addison-Wesley: Boston, MA, USA, 1989.
4. Whitley, D. A genetic algorithm tutorial. *Stat. Comput.* **1994**, *4*, 65–85. [CrossRef]
5. Tsang, P.W.M.; Au, A.T.S. A genetic algorithm for projective invariant object recognition. *IEEE TENCON Digit. Signal Process. Appl.* **1996**, *1*, 58–63.
6. Mohammed, A.A.; Nagib, G. Optimal routing in ad-hoc network using genetic algorithm. *Int. J. Adv. Netw. Appl.* **2012**, *03*, 1323–1328.
7. Srivastava, P.R.; Kim, T.H. Application of genetic algorithm in software testing. *Int. J. Softw. Eng. Its Appl.* **2009**, *3*, 87–96.

8. Paulinas, M.; Ušinskas, A. A survey of genetic algorithms applications for image enhancement and segmentation. *Inf. Technol. Control* **2015**, *36*, 278–284.
9. Hassanat, A.B.; Prasath, V.S.; Mseidein, K.I.; Al-Awadi, M.; Hammouri, A.M. A HybridWavelet-Shearlet Approach to Robust Digital ImageWatermarking. *Informatica* **2017**, *41*, 3–24.
10. Benkhellat, Z.; Belmehdi, A. Genetic algorithms in speech recognition systems. In Proceedings of the International Conference on Industrial Engineering and Operations Management, Istanbul, Turkey, 3–6 July 2012; pp. 853–858.
11. Gupta, H.; Wadhwa, D.S. Speech feature extraction and recognition using genetic algorithm. *Int. J. Emerg.* **2014**, *4*, 363–369.
12. Aliakbarpour, H.; Prasath, V.B.S.; Dias, J. On optimal multi-sensor network configuration for 3D registration. *J. Sens. Actuator Netw.* **2015**, *4*, 293–314. [CrossRef]
13. Papanna, N.; Reddy, A.R.M.; Seetha, M. EELAM: Energy efficient lifetime aware multicast route selection for mobile ad hoc networks. *Appl. Comput. Inform.* **2019**, *15*, 120–128. [CrossRef]
14. Ashish, K.; Dasari, A.; Chattopadhyay, S.; Hui, N.B. Genetic-neuro-fuzzy system for grading depression. *Appl. Comput. Inform.* **2018**, *14*, 98–105. [CrossRef]
15. Omisore, M.O.; Samuel, O.W.; Atajeromavwo, E.J. A Genetic-Neuro-Fuzzy inferential model for diagnosis of tuberculosis. *Appl. Comput. Inform.* **2017**, *13*, 27–37. [CrossRef]
16. Hassanat, A.B.; Altarawneh, G.; Tarawneh, A.S.; Faris, H.; Alhasanat, M.B.; de Voogt, A.; Prasath, S.V. On Computerizing the Ancient Game of *ṭāb*. *Int. J. Gaming Comput.-Mediat. Simul. (IJGCMS)* **2018**, *10*, 20–40. [CrossRef]
17. Guo, K.; Yang, M.; Zhu, H. Application research of improved genetic algorithm based on machine learning in production scheduling. In *Neural Computing and Applications*; Springer: London, UK, 2019; pp. 1–12.
18. Wang, Y.M.; Yin, H.L. Cost-Optimization Problem with a Soft Time Window Based on an Improved Fuzzy Genetic Algorithm for Fresh Food Distribution. *Math. Probl. Eng.* **2018**, *2018*, 1–16. [CrossRef]
19. Hendricks, D.; Wilcox, D.; Gebbie, T. High-speed detection of emergent market clustering via an unsupervised parallel genetic algorithm. *arXiv* **2014**, arXiv:1403.4099.
20. Mustafa, W. Optimization of production systems using genetic algorithms. *Int. J. Comput. Intell. Appl.* **2003**, *3*, 233–248. [CrossRef]
21. Eiben, A.E.; Smith, J.E. *Introduction to Evolutionary Computing*; Springer: Berlin/Heidelberg, Germany, 2003.
22. Zhong, J.; Hu, X.; Gu, M.; Zhang, J. Comparison of performance between different selection strategies on simple genetic algorithms. In Proceedings of the International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC'06), Vienna, Austria, 28–30 November 2005; Volume 2, pp. 1115–1121.
23. Eiben, A.E.; Michalewicz, Z.; Schoenauer, M.; Smith, J.E. Parameter control in evolutionary algorithms. In *Parameter Setting in Evolutionary Algorithms*; Springer: Berlin/Heidelberg, Germany, 2007; pp. 19–46.
24. Hong, T.P.; Hong-Shung, W. A dynamic mutation genetic algorithm. *IEEE Int. Conf. Syst. Man, Cybern.* **1996**, *3*, 2000–2005.
25. Gen, M.; Runwei, C. *Genetic Algorithms and Engineering Optimization*; Wiley: Hoboken, NJ, USA, 2000.
26. Deb, K.; Agrawal, S. Understanding interactions among genetic algorithm parameters. *Found. Genet. Algorithms* **1999**, *5*, 265–286.
27. Pereira, G.C.; Oliveira, M.M.F.; Ebecken, N.F.F. Genetic Optimization of Artificial Neural Networks to Forecast Virioplankton Abundance from Cytometric Data. *J. Intell. Learn. Syst. Appl.* **2013**, *5*, 57–66. [CrossRef]
28. Kumar, R. Novel encoding scheme in genetic algorithms for better fitness. *Int. J. Eng. Adv. Technol.* **2012**, *1*, 214–219.
29. Shyr, W.J. *Parameters Determination for Optimum Design by Evolutionary Algorithm*; IntechOpen: Rijeka, Croatia, 2010.
30. Alkafaween, E.O. Novel Methods for Enhancing the Performance of Genetic Algorithms. Master's Thesis, Mu'tah University, Karak, Jordan, 2015.
31. Shukla, A.; Pandey, H.M.; Mehrotra, D. Comparative review of selection techniques in genetic algorithm. In Proceedings of the International Conference on Futuristic Trends on Computational Analysis and Knowledge Management, Noida, India, 25–27 February 2015; pp. 515–519.

32. Grefenstette, J. Optimization of control parameters for genetic algorithms. *IEEE Trans. Syst. Man Cybern.* **1986**, *16*, 122–128. [CrossRef]
33. Razali, N.M.; Geraghty, J. Genetic algorithm performance with different selection strategies in solving TSP. In Proceedings of the world congress on engineering, London, UK, 6–8 July 2011; pp. 1–6.
34. Oladele, R.O.; Sadiku, J.S. Genetic algorithm performance with different selection methods in solving multi-objective network design problem. *Int. J. Comput. Appl.* **2013**, *70*, 5–9. [CrossRef]
35. Bäck, T. *Evolutionary Algorithms in Theory and Practice*; Oxford University Press: Oxford, UK, 1996.
36. Lipowski, A.; Lipowska, D. Roulette-wheel selection via stochastic acceptance. *Phys. A Stat. Mech. Its Appl.* **2012**, *391*, 2193–2196. [CrossRef]
37. Srinivas, M.; Patnak, L.M. Adaptive probabilities of crossover and mutation in genetic algorithms. *IEEE Trans. Syst. Man Cybern.* **1994**, *24*, 656–667. [CrossRef]
38. Obitko, M. *Introduction to Genetic Algorithms*; Czech Technical University: Prague, Czech Republic, 1998.
39. Kaya, Y.; Uyar, M. A novel crossover operator for genetic algorithms: Ring crossover. *arXiv* **2011**, arXiv:1105.0355.
40. Bajpai, P.; Manoj, K. Genetic Algorithm—An Approach to Solve Global Optimization. *Indian J. Comput. Sci. Eng.* **2010**, *1*, 199–206.
41. Korejo, I.; Yang, S.; Brohi, K.; Khuhro, Z.U. Multi-population methods with adaptive mutation for multi-modal optimization problems. *Int. J. Soft Comput. Artif. Intell. Appl.* **2013**, *2*, 19. [CrossRef]
42. Safe, M.; Carballido, J.; Ponzoni, I.; Brignole, N. On stopping criteria for genetic algorithms. In *Brazilian Symposium on Artificial Intelligence*; Springer: Berlin/Heidelberg, Germany, 2004; pp. 405–413.
43. De Jong, K.; Spears, W. A formal analysis of the role of multi-point crossover. *Ann. Math. Artif. Intell.* **1992**, *5*, 1–26. [CrossRef]
44. Lynch, M. Evolution of the mutation rate. *Trends Genet.* **2010**, *26*, 345–352. [CrossRef]
45. Roeva, O.; Fidanova, S.; Paprzycki, M. Influence of the population size on the genetic algorithm performance in case of cultivation process modelling. In Proceedings of the IEEE Conference on Computer Science and Information Systems, Kraków, Poland, 8–11 September 2013; pp. 371–376.
46. Tuson, A.L.; Ross, P. Adapting Operator Probabilities in Genetic Algorithms. Master's Thesis, Department of Artificial Intelligence, Univeristy of Edinburgh, Edinburgh, UK, 1995.
47. Davis, L. *Handbook of Genetic Algorithms*; Van Nostrand Reinhold Co.: New York, NY, USA, 1991.
48. Munroe, R.D. Genetic programming: The ratio of crossover to mutation as a function of time. *Res. Lett. Inf. Math. Sci.* **2004**, *6*, 83–96.
49. Bernard, T. *Modeling, Control and Optimization of Water System*; Springer: Berlin/Heidelberg, Germany, 2016.
50. Eiben, A.E.; Hinterding, R.; Michalewicz, Z. Parameter control in evolutionary algorithms. *IEEE Trans. Evol. Comput.* **1999**, *3*, 124–141. [CrossRef]
51. Beasley, D.; Martin, R.R.; Bull, D.R. An overview of genetic algorithms: Part 1 Fundamentals. *Univ. Comput.* **1993**, *15*, 56–69.
52. Beasley, D.; Bull, D.R.; Martin, R.R. An overview of genetic algorithms: Part 2 Research topics. *Univ. Comput.* **1993**, *15*, 170–181.
53. Srinivas, M.; Patnaik, L.M. Genetic algorithms: A survey. *Computer* **1994**, *27*, 17–26. [CrossRef]
54. DeJong, K. Analysis of the Behavior of a Class of Genetic Adaptive. Ph.D. Thesis, University of Michigan, Ann Arbor, MI, USA, 1975.
55. Schlierkamp-Voosen, D. Optimal interaction of mutation and crossover in the breeder genetic algorithm. In *International Conference on Genetic Algorithms*; Morgan Kaufmann Publishers Inc.: San Francisco, CA, USA, 1993; 648p.
56. Hong, T.P.; Wang, H.S.; Lin, W.Y.; Lee, W.Y. Evolution of appropriate crossover and mutation operators in a genetic process. *Appl. Intell.* **2002**, *16*, 7–17. [CrossRef]
57. Pelikan, M.; Goldberg, D.E.; Cantú-Paz, E. Bayesian optimization algorithm, population sizing, and time to convergence. In Proceedings of the Annual Conference on Genetic and Evolutionary Computation, Las Vegas, NV, USA, 10–12 July 2000; pp. 275–282.
58. Piszcz, A.; Soule, T. Genetic programming: Optimal population sizes for varying complexity problems. In Proceedings of the Annual Conference on Genetic and Evolutionary Computation, Seattle, WA, USA, 8–12 July 2006; pp. 953–954.

59. Katsaras, V.; Koumousis, C.P. A sawtooth genetic algorithm combining the effects of variable population size and reinitialization to enhance performance. *IEEE Trans. Evol. Comput.* **2006**, *10*, 19–28.
60. Lobo, G.; Goldberg, F. The parameterless genetic algorithm in practice. *Inf. Sci.* **2004**, *167*, 217–232. [CrossRef]
61. Gotshall, S.; Rylander, B. Optimal population size and the genetic algorithm. *Population* **2002**, *100*, 900–904.
62. Diaz-Gomez, P.A.; Hougen, D.F. Initial population for genetic algorithms: A metric approach. *GEM* **2007**, 43–49. Available online: http://www.cameron.edu/pdiaz-go/GAsPopMetric.pdf (accessed on 9 December 2019).
63. Dong, M.; Wu, Y. Dynamic crossover and mutation genetic algorithm based on expansion sampling. In *International Conference on Artificial Intelligence and Computational Intelligence*; Springer: Berlin/Heidelberg, Germany, 2009; pp. 141–149.
64. Abu Alfeilat, H.A.; Hassanat, A.B.; Lasassmeh, O.; Tarawneh, A.S.; Alhasanat, M.B.; Eyal, Salman, H.S.; Prasath, V.S. Effects of Distance Measure Choice on K-Nearest Neighbor Classifier Performance: A Review. *Big Data* **2019**, *7*, 1–28. [CrossRef]
65. Chiroma, H.; Abdulkareem, S.; Abubakar, A.; Zeki, A.; Gital, A.Y.; Usman, M.J. Correlation study of genetic algorithm operators: Crossover and mutation probabilities. In Proceedings of the International Symposium on Mathematical Sciences and Computing Research, Ipoh City, Malaysia, 6–7 December 2013; pp. 39–43.
66. Huang, Y.; Shi, K. Genetic algorithms in the identification of fuzzy compensation system. In Proceedings of the 1996 IEEE International Conference on Systems, Man and Cybernetics. Information Intelligence and Systems (Cat. No.96CH35929), Beijing, China, 14–17 Octember 1996; pp. 1090–1095.
67. Renders, J.; Flasse, S.P. Hybrid methods using genetic algorithms for global optimization. *IEEE Trans. Syst. Man, Cybern. Part B Cybern.* **1996**, *26*, 243–258. [CrossRef] [PubMed]
68. Vavak, F.; Fogarty, C. Comparison of steady state and generational genetic algorithms for use in nonstationary environments. In Proceedings of the IEEE International Conference on Evolutionary Computation, Nagoya, Japan, 20–22 May 1996; pp. 192–195.
69. Chaiyaratana, N.; Zalzala, A. Hybridisation of neural networks and genetic algorithms for time-optimal control. In Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406), Washington, DC, USA, 6–9 July 1999.
70. Man, F.K.; Tang, S.K.; Kwong, S. *Genetic Algorithms: Concepts and Designs*; Springer: Berlin/Heidelberg, Germany, 1999.
71. Ammar, H.H.; Tao, Y. Fingerprint registration using genetic algorithms. In Proceedings of the 3rd IEEE Symposium on Application-Specific Systems and Software Engineering Technology, Richardson, TX, USA, 24–25 March 2000; pp. 148–154.
72. Jareanpon, C.; Pensuwon, W.; Frank, R.J.; Dav, N. An adaptive RBF network optimised using a genetic algorithm applied to rainfall forecasting. In Proceedings of the IEEE International Symposium on Communications and Information Technology, Sapporo, Japan, 26–29 October 2004; pp. 1005–1010.
73. Yu, X.; Meng, B. Research on dynamics in group decision support systems based on multi-objective genetic algorithms. In Proceedings of the IEEE International Conference on Service Systems and Service Management, Troyes, France, 25–27 October 2006; pp. 895–900.
74. Alexandre, E.; Cuadra, L.; Rosa, M. Feature selection for sound classification in hearing aids through restricted search driven by genetic algorithms. *IEEE Trans. Audio Speech Lang. Process.* **2007**, *15*, 2249–2256. [CrossRef]
75. Meng, X.; Song, B. Fast genetic algorithms used for PID parameter optimization. In Proceedings of the IEEE International Conference on Automation and Logistics, Jinan, China, 18–21 August 2007; pp. 2144–2148.
76. Krawiec, K. Generative learning of visual concept using multi objective genetic programming. *Pattern Recognit. Lett.* **2007**, *28*, 2385–2400. [CrossRef]
77. Hamdan, M. A heterogeneous framework for the global parallelization of genetic algorithms. *Int. Arab J. Inf. Technol.* **2008**, *5*, 192–199.
78. Liu, J. Application of fuzzy neural networks based on genetic algorithms in integrated navigation system. In Proceedings of the IEEE Conference on Intelligent Computation Technology and Automation, Changsha, China, 10–11 October 2009; pp. 656–659.
79. Ka, Y.W.; Chi, L. Positioning weather systems from remote sensing data using genetic algorithms. In *Computational Intelligence for Remote Sensing*; Springer: Berlin/Heidelberg, Germany, 2008; pp. 217–243.

80. Sorsa, A.; Peltokangas, R.; Leiviska, K. Real coded genetic algorithms and nonlinear parameter identification. In Proceedings of the IEEE International Conference on Intelligent System, Varna, Bulgaria, 6–8 September 2009; pp. 42–47.
81. Ai, J.; Feng-Wen, H. Methods for optimizing weights of wavelet neural network based on adaptive annealing genetic algorithms. In Proceedings of the IEEE International Conference on Industrial Engineering and Engineering Management, Beijing, China, 21–23 October 2009; pp. 1744–1748.
82. Krömer, P.; Platoš, J.; Snášel, V. Modeling permutation for genetic algorithms. In Proceedings of the IEEE International Conference on Soft Computing Pattern Recognition, Malacca, Malaysia, 4–7 December 2009; pp. 100–105.
83. Lin, C. An adaptive genetic algorithms based on population diversity strategy. In Proceedings of the IEEE International Conference on Genetic and Evolutionary Computing, Guilin, China, 14–17 October 2009; pp. 93–96.
84. Lizhe, Y.; Bo, X.; Xiangjie, W. BP network model optimized by adaptive genetic algorithms and the application on quality evaluation for class. In Proceedings of the IEEE International Conference on Future Computer and Communication, Wuhan, China, 21–24 May 2010; p. 273.
85. Belevičius, R.; Ivanikovas, S.; Šešok, D.; Valenti, S. Optimal placement of piles in real grillages: Experimental comparison of optimization algorithms. *Inf. Technol. Control* **2011**, *40*, 123–132.
86. Zhang, L.; Zhang, X. Measurement of the optical properties using genetic algorithms optimized neural networks. In Proceedings of the 2011 Symposium on Photonics and Optoelectronics (SOPO), Wuhan, China, 16–18 May 2011; pp. 1–4.
87. Laboudi, Z.; Chikhi, S. Comparison of genetic algorithms and quantum genetic algorithms. *Int. Arab J. Inf. Technol.* **2012**, *7*, 243–249.
88. Capraro, C.T.; Bradaric, I.; Capraro, G.; Kong, L. Using genetic algorithms for radar waveform selection. In Proceedings of the 2008 IEEE Radar Conference, Rome, Italy, 26–30 May 2008; pp. 1–6.
89. Guo, F.; Qi, M. Application of Elman neural network based on improved niche adaptive genetic algorithm. *IEEE Int. Conf. Intell. Control Inf. Process.* **2011**, *2*, 660–664.
90. Laporte, G. The traveling salesman problem: An overview of exact and approximate algorithms. *Eur. J. Oper. Res.* **1992**, *59*, 231–247. [CrossRef]
91. 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*, 785–801.
92. Hassanat, A.B.; Alkafaween, E. On enhancing genetic algorithms using new crossovers. *Int. J. Comput. Appl. Technol.* **2017**, *55*, 202–212. [CrossRef]
93. Potvin, J.Y. Genetic algorithms for the traveling salesman problem. *Ann. Oper. Res.* **1996**, *63*, 337–370. [CrossRef]
94. Reisleben, B.; Merz, P. A genetic local search algorithm for solving symmetric and asymmetric traveling salesman problems. In Proceedings of the IEEE International Conference on Evolutionary Computation, Nagoya, Japan, 20–22 May 1996; pp. 616–621.
95. Larrañaga, P.; Kuijpers, C.M.; Murga, R.H.; Inza, I.; Dizdarevic, S. Genetic algorithms for the travelling salesman problem: A review of representations and operators. *Artif. Intell. Rev.* **1999**, *13*, 129–170. [CrossRef]
96. Soni, N.; Kumar, T. Study of various mutation operators in genetic algorithms. *Int. J. Comput. Sci. Inf. Technol.* **2014**, *5*, 4519–4521.
97. Hassanat, A.; Prasath, V.; Abbadi, M.; Abu-Qdari, S.; Faris, H. An improved genetic algorithm with a new initialization mechanism based on regression techniques. *Information* **2018**, *9*, 167. [CrossRef]
98. Kaabi, J.; Harrath, Y. Permutation rules and genetic algorithm to solve the traveling salesman problem. *Arab J. Basic Appl. Sci.* **2019**, *26*, 283–291. [CrossRef]
99. Gendreau, M.; Hertz, A.; Laporte, G. A tabu search heuristic for the vehicle routing problem. *Manag. Sci.* **1994**, *40*, 1276–1290. [CrossRef]
100. Dorigo, M.; Gambardella, L.M. Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE Trans. Evol. Comput.* **1997**, *1*, 53–66. [CrossRef]
101. Shi, X.H.; Liang, Y.C.; Lee, H.P.; Lu, C.; Wang, Q.X. Particle swarm optimization-based algorithms for TSP and generalized TSP. *Inf. Process. Lett.* **2007**, *103*, 169–176. [CrossRef]

102. Malek, M.; Guruswamy, M.; Pandya, M.; Owens, H. Serial and parallel simulated annealing and tabu search algorithms for the traveling salesman problem. *Ann. Oper. Res.* **1989**, *21*, 59–84. [CrossRef]
103. Aarts, E.H.L.; Stehouwer, H.P. Neural networks and the travelling salesman problem. In Proceedings of the International Conference on Artificial Neural Networks, Amsterdam, The Netherlands, 13–16 September 1993; pp. 950–955.
104. Durbin, R.; Szeliski, R.; Yuille, A. An analysis of the elastic net approach to the traveling salesman problem. *Neural Comput.* **1989**, *1*, 348–358. [CrossRef]
105. Karkory, F.A.; Abudalmola, A.A. Implementation of heuristics for solving travelling salesman problem using nearest neighbour and minimum spanning tree algorithms. *Int. J. Math. Comput. Phys. Electr. Comput. Eng.* **2013**, *7*, 1524–1534.
106. Singh, A.; Singh, R. Exploring travelling salesman problem using genetic algorithm. *Int. J. Eng. Res. Technol.* **2014**, *3*, 2032–2035.
107. Ahmed, Z.H. Genetic algorithm for the traveling salesman problem using sequential constructive crossover operator. *Int. J. Biom. Bioinform.* **2010**, *3*, 96–105.
108. Banzhaf, W. The "molecular" traveling salesman. *Biol. Cybern.* **1990**, *64*, 7–14. [CrossRef]
109. Davis, L. Applying adaptive algorithms to epistatic domains. *Int. Jt. Conf. Artif. Intell.* **1985**, *85*, 162–164.
110. Abdoun, O.; Abouchabaka, J.; Tajani, C. Analyzing the performance of mutation operators to solve the travelling salesman problem. *Int. J. Emerg. Sci.* **2012**, *2*, 61–77.
111. Reinelt, G. TSPLIB—A traveling salesman problem library. *ORSA J. Comput.* **1991**, *3*, 376–384. [CrossRef]
112. Hassanat, A. Furthest-Pair-Based Decision Trees: Experimental Results on Big Data Classification. *Information* **2018**, *9*, 284. [CrossRef]
113. Hassanat, A. Norm-Based Binary Search Trees for Speeding Up KNN Big Data Classification. *Computers* **2018**, *7*, 54. [CrossRef]
114. Hassanat, A.B. Two-point-based binary search trees for accelerating big data classification using KNN. *PLoS ONE* **2018**, *13*, e0207772. [CrossRef]
115. Hassanat, A.B. Furthest-pair-based binary search tree for speeding big data classification using k-nearest neighbors. *Big Data* **2018**, *6*, 225–235. [CrossRef]

Sample Availability: TSP Data sets used in this work are available from the TSPLIB: http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsp/index.html.