Experimental Study of Hybrid Genetic Algorithms for the Maximum Scatter Travelling Salesman Problem

Zakir Hussain Ahmed[1]*, Asaad Shakir Hameed[2]
Modhi Lafta Mutar[3], Mohammed F. Alrifaie[4], Mundher Mohammed Taresh[5]
Department of Mathematics and Statistics, College of Science[1]
Imam Mohammad Ibn Saud Islamic University (IMSIU), Riyadh, Kingdom of Saudi Arabia[1]
Department of Mathematics, General Directorate of Thi-Qar Education, Ministry of education, Thi-Qar, Iraq[2,3]
[4]Department of Information and Communications, Basra University College of science and technology, Basrah, Iraq[4]
College of Information Science and Engineering, Hunan University, Chang Sha, China[5]

***Abstract*—We consider the maximum scatter travelling salesman problem (MSTSP), a travelling salesman problem (TSP) variant. The problem aims to maximize the shortest edge in the tour that travels each city only once in the given network. It is a very complicated NP-hard problem, and hence, exact solutions are obtainable for small sizes only. For large sizes, heuristic algorithms must be applied, and genetic algorithms (GAs) are observed to be very successful in dealing with such problems. In our study, a simple GA (SGA) and four hybrid GAs (HGAs) are proposed for the MSTSP. The SGA starts with initial population produced by sequential sampling approach that is improved by 2-opt search, and then it is tried to improve gradually the population through a proportionate selection procedure, sequential constructive crossover, and adaptive mutation. A stopping condition of maximum generation is adopted. The hybrid genetic algorithms (HGAs) include a selected local search and perturbation procedure to the proposed SGA. Each HGA uses one of three local search procedures based on insertion, inversion and swap operators directly or randomly. Experimental study has been carried out among the proposed SGA and HGAs by solving some TSPLIB asymmetric and symmetric instances of various sizes. Our computational experience reveals that the suggested HGAs are very good. Finally, our best HGA is compared with a state-of-art algorithm by solving some TSPLIB symmetric instances of many sizes. Our computational experience reveals that our best HGA is better.**

Keywords*—Hybrid genetic algorithm; maximum scatter travelling salesman problem; sequential constructive crossover; adaptive mutation; local search; perturbation procedure

I. Introduction

The travelling salesman problem (TSP) is a popular problem, which finds smallest tour of the salesman that starts journey from a headquarters city and visits all outstanding n cities (nodes) exactly once before comes back to his headquarters. The TSP is NP- Hard [1] and several good procedures are suggested to solve the problem. However, some circumstances need different constraints to accept a tour as solution. One such constraint is to maximize the shortest edge in the tour, and the TSP having such constraint is called the maximum scatter TSP (MSTSP). So, the MSTSP finds a Hamiltonian cycle/circuit so as to maximize the shortest edge. That means, each city in the Hamiltonian circuit is far from (scattered) its preceding and succeeding cities. The problem is also known as the max-min 1-neighbour TSP. In general, the max-min m-neighbour TSP aims to maximize the shortest edge (distance) between a city and its all m-neighbor cities in the Hamiltonian cycle/circuit. The bottleneck TSP (BTSP) is very close to the MSTSP. The BTSP aims to minimize the longest edge [2]. Further, the maximum TSP (MaxTSP) which finds a Hamiltonian cycle/circuit to maximize the length of any tour is also closely related to the MSTSP [3]. Fig. 1 shows the difference between TSP and MSTSP on an instance of 29 cities [4]. It is clear from the figure that the TSP aims to decrease the total distance covered by the salesman, whereas the MSTSP aims to maximize the shortest edge by producing any two successive cities in its tour as much scattered as possible.

Let us formally define the MSTSP as follows: Let a network with n cities (city 1 is the headquarters) and an nXn distance (time or cost, etc.) matrix D=[d_{ij}] associated with ordered pair (i, j) of cities is given. Let $(1=\alpha_0, \alpha_1, \alpha_2, ..., \alpha_{n-1}, \alpha_n=1) \equiv \{1\rightarrow\alpha_1\rightarrow\alpha_2\rightarrow .... \rightarrow\alpha_{n-1}\rightarrow 1\}$ be a tour. The tour value is defined as $\min\{d_{\alpha_i,\alpha_{i+1}}: i = 0, 1, 2, ..., n-1\}$. The objective is to maximize the tour value.

The problem may be converted to the BTSP by supposing c_{ij} = M-d_{ij}, where C = [c_{ij}] is the corresponding BTSP's distance (or cost) matrix and M is a big number [5]. The MSTSP was first defined in [6], which has several applications ([1], [7]). The MSTSP is NP-hard [1], and no polynomial-time algorithm is available for solving the problem. So, finding optimal solution for large-sized problem instances using exact method is not possible. Thus, for finding better solution, within acceptable computational effort, to such type of problems, generally, heuristic/metaheuristic algorithms are applied. Tabu search [8], simulated annealing [9], ant colony algorithm [10], insertion heuristic [11], variable neighbourhood method [12], discrete differential evolution algorithm [13], genetic algorithms [14], etc., are some popular metaheuristic algorithms. Among them, genetic algorithms (GAs) are widely used algorithms, and so, we are using GAs to solve the MSTSP.

GAs are based on simulating the Darwinian survival-of-the-fittest theory in the environmental biology [14]. They are very robust, parallel, and global search metaheuristics that can solve large-sized problems quickly. They can automatically obtain and collect knowledge throughout the search procedure

*Corresponding Author

and can adaptively manage the search procedure to obtain the optimal/best solution. They were effectively applied to various complex optimization problems for solving them. For any problem, each feasible solution may be encoded as a string called chromosome or individual whose value is its objective function [15]. Chromosomes are collections of genes.

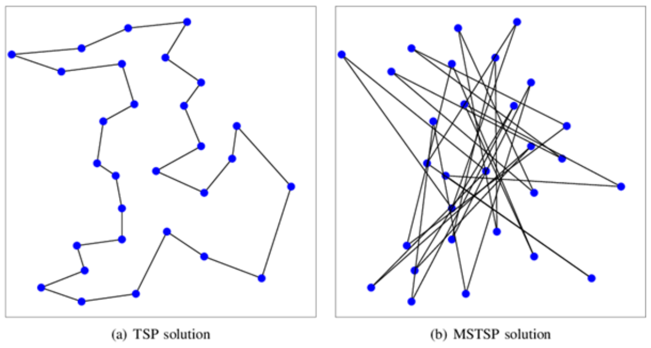

Fig. 1. Difference between TSP and MSTSP.

Simple GAs start from a chromosome set known as initial population and then go through mainly three basic operators – selection, crossover, and mutation, to generate improved populations in following generations. Selection operator probabilistically copies some chromosomes to the following generation. Crossover arbitrarily selects two parent chromosomes and mates them to produce offspring chromosome(s). Mutation picks out a position at a chromosome randomly and changes its value. The crossover along with selection is the major procedure in GAs. Mutation varies the search space and defends genetic material losses. Thus, crossover probability is set to be very high, whereas mutation probability is set to be very low [14]. As crossover operator is very important operator, so, using better crossover operators can achieve better GAs. Normally, crossover methods that were applied for the usual TSP are applied to its variations also. A computational experience carried amongst eight crossover methods for the MSTSP proven that sequential constructive crossover is the best operator [16].

Though simple GAs using three basic operators can solve complex optimization problems quickly, but very often they converge prematurely, and get trapped in local minima [17]. So, one must apply some techniques to overcome premature convergence issue and to enhance the solution obtained by simple GAs. So, this paper develops a simple GA (SGA) and four hybrid GAs (HGA1, HGA2, HGA3 and HGA4) for finding solution to the MSTSP. Our proposed SGA uses sequential sampling algorithm along with 2-opt search for initial population generation, sequential constructive crossover, and adaptive mutation. The hybrid genetic algorithms (HGAs) include a selected local search and pertu`rbation procedure to the proposed SGA. Each HGA uses one of the local search procedures based on insertion, inversion, and swap operators. Generally, perturbation procedure is used to overcome premature convergence issue. The partially mapped crossover [15] along with swap mutation for perturbation procedure is to find better quality solution to the MSTSP.

The usefulness of our proposed HGAs have been examined amongst themselves and calculated percentage of improvement of the obtained solution over the solution obtained by SGA for the asymmetric and symmetric TSPLIB problem instances. The experimental results show a very good improvement of the solutions by HGAs over the solutions by SGA. Further, it is seen that for asymmetric instances, HGA3 is placed in 2nd position and HGA4 is the best one. For symmetric instances, HGA2 is placed in 2nd position and HGA4 is the best one. Overall, for both categories of the instances, HGA4 is the best one, HGA2 is the 2nd best and HGA3 is the 3rd best. Finally, our HGA4 is compared against multi-start iterated local search (MS-ILS(h1+h2)) [4] by solving some TSPLIB symmetric instances of different sizes. Our computational experience reveals that our HGA4 is better than MS-ILS(h1+h2).

This paper is arranged as follows: A literature survey for the MSTSP is provided in Section II. Section III develops simple and hybrid GAs for the problem, while Section IV reports computational experience of the developed algorithms. Finally, Section V provides conclusion and forthcoming research works.

II. Literature Review

The MSTSP is a difficult NP-hard problem. Methods to solve this kind of optimization problems are grouped into two broad groups – exact and heuristic methods ([18]-[19]). There are very less literatures on the MSTSP. The first procedure for solving the problem is developed by Arkin et al. [1]. They proved that the MSTSP is NP-hard, and no constant-factor approximation procedure can be devised unless NP = P. A factor-2 (claimed to be best) approximation procedure is developed for max-min 1-neighbour TSP satisfying the triangle inequality for path and cycle adaptations. Further, they developed procedures for the max-min 2-neighbour TSP satisfying triangle inequality for cycle and path adaptations. Finally, the procedures extended to find an approximation solution of the max-min m-neighbour TSP for path version.

Approximation procedures for max-min 2-neighbour TSP with triangle inequality was developed by Chiang [20] for the cycle and path adaptations by improving the procedures in [1]. As reported, both procedures are very simple. Some studies on the MSTSP and its related versions are reported by John [7].

An approximation procedure for the MSTSP satisfying triangle inequality was developed by Kabadi and Punnen [21] that claimed to find the best bound for this case.

An improved procedure of the procedure in [1] was proposed for the points on a line to a regular mXn-grid by Hoffmann et al. [22] that claimed to obtain optimal solutions. They further claimed that the procedure takes linear computational effort to obtain optimal tour in some cases.

The multi-salesmen MSTSP called multiple MSTSP (MMSTSP) was proposed by Dong et al. [23]. They proposed three improved GAs for the problem. Their improved algorithms used greedy initialization, simulated annealing, and hill-climbing algorithms. As reported, their algorithms are effective algorithms that can expose various characteristics to find solution of the problem.

In [4], a multi-start iterated local search procedure was developed for the MSTSP. Based on modified 2-opt moves and insertion, two local search procedures were proposed in their procedure. As reported, their algorithm found very good results on some symmetric TSPLIB instances.

In [16], eight GAs were developed using eight crossover methods for the MSTSP. A comparative study was reported on some TSPLIB asymmetric and symmetric instances. It was showed that the sequential constructive crossover (SCX) is the best, partially mapped crossover (PMX) is the second-best and greedy crossover (GX) is the worst.

It is mentioned that the BTSP is very close to the MSTSP. Lexisearch approaches were developed for the BTSP in ([24], [25]). Further hybrid algorithms were developed for the BTSP in ([26], [27]). The MaxTSP is also close to the MSTSP for which a hybrid GA is developed for finding solution to the problem [28].

Since there are a few literature on the hybrid algorithms on the MSTSP, hence we propose to develop hybrid genetic algorithms to show the efficiency of the hybrid algorithms in solving the problem.

III. Hybrid Genetic Algorithms for the MSTSP

Genetic algorithms (GAs) are established to be effective for the traditional TSP and its some variants. Though they do not assure the optimality of their obtained solutions, they normally obtain very close optimal solutions rapidly. In this section, we develop a simple GA (SGA) and four hybrid GAs (HGAs) for the MSTSP.

A. Chromosome Representation

The first job in GAs is to determine a chromosome representation procedure for representing solutions of a problem so that GA operators can produce feasible chromosome(s). For TSP and its variants, mainly path representation is used which lists cities so that no city is duplicated in a chromosome. We consider this path representation for the MSTSP. As an example, let {1, 2, 3, 4, 5, 6, 7, 8} be the cities in an 8-city problem, and the chromosome (1, 3, 2, 7, 8, 6, 4, 5) denotes the tour $\{1\rightarrow3\rightarrow2\rightarrow7\rightarrow8\rightarrow6\rightarrow4\rightarrow5 \rightarrow1\}$ whose objective as well as fitness function is the shortest edge in this tour. As MSTSP is a maximization problem, a higher fitness value is better than the lower fitness value.

B. Improved Initial Population

Starting with an improved initial population can provide good solutions quickly. We use sequential sampling approach [26] for generating initial population for our GAs, that was successfully applied on other TSP variants ([27]-[28]). In sequential sampling approach, first alphabet table is constructed based on the given distance (cost) matrix, then the probability of visiting every un-visited city is allocated in each row such that first un-visited city is allocated higher probability than probability of 2nd city, then 2nd one is allocated higher than the 3rd city, and so forth. For each un-visited city in that row, cumulative probability is also calculated. The city is accepted that represents a randomly generated number in a cumulative probability interval. This process is repeated until a valid chromosome is created. This way, a population of given size is generated. However, it is observed that this approach cannot search all space. So, to improve the initial population, we apply 2-opt search to every chromosome for enhancing the population. However, if the newly obtained chromosome is better than the old one, replace it by the new one, otherwise, no action is taken. Due to the strong capability of 2-opt local search, it can improve the search space of our proposed algorithm.

C. Selection Strategy

The selection strategy is the procedure of choosing parents from the current population for the next operation. In selection operation, no new chromosome is created, only some of the fitter chromosomes are passed to the breeding pool for the subsequent operation/generation. By selecting a greater section of fitter chromosomes, this operation simulates the Darwinian hypothesis of survival-of-the-fittest in biology. Normally, the proportionate selection is used where a chromosome is chosen depending on its probability of selection. We use stochastic remainder selection procedure [29] for the proposed GAs. In this procedure, first ‘expected count’ of every individual is computed by dividing their corresponding fitness value with the average fitness value. Then as many individuals are copied equal to the mantissa of the expected counts, and then mantissas are subtracted from the corresponding expected counts. This will result the values of the expected counts less than one. If a randomly generated number is less than the expected count of a selected individual, then the individual is inserted into the mating pool. Repeat this procedure till the number of chromosomes is equal to the size of population. Note that population size is the number of chromosomes in the population.

D. Crossover Operator

Crossover operator performs a very big role in GAs, where two parent chromosomes as well as a crossover point within the chromosomes' length are chosen and the chromosomes’ data after the crossover point are exchanged. Quite a few good crossover methods are available in the literature for traditional TSP that might be applied for the MSTSP. Ahmed [16] applied eight crossover operators, namely, ordered crossover [30], partially mapped crossover [31], cycle crossover [32], alternating edges crossover [33], generalized N crossover [34], greedy crossover [33], edge recombination crossover [35], sequential constructive crossover [15] on the MSTSP, and reported a comparative study among them. As reported, sequential constructive crossover (SCX) is observed as the best method. We also apply this SCX in our proposed GAs. The steps of SCX algorithm are as follows [16]:

Step 1: Start from 'city 1’ (i.e., current city p =1).

Step 2: Search sequentially both parent chromosomes and take the first un-visited city emerged after 'city p’ in the parents. If no un-visited city after 'city p’ is available in a parent chromosome, search from beginning of the chromosome and take the first un-visited city and go to Step 3.

Step 3: Suppose 'city α' and 'city β' are in 1st and 2nd parents correspondingly, then for choosing the following city go to Step 4.

Step 4: If $d_{p\alpha} > d_{p\beta}$, then choose 'city α', otherwise, 'city β' as subsequent city and merge it to the current offspring. If the offspring is a full chromosome, stop, else, the current city is renamed as 'city p', go to Step 2.

Let us illustrate the SCX using a 7-city instance with distance matrix provided in Table I. Let P_1: (1, 5, 3, 2, 7, 4, 6) and P_2: (1, 5, 7, 3, 6, 2, 4) be parent chromosomes with costs 3 and 2 respectively. Our computation is started from the city 1 (headquarters).

TABLE I. THE DISTANCE MATRIX

City	**1**	**2**	**3**	**4**	**5**	**6**	**7**
1	0	7	15	9	10	6	8
2	11	0	8	7	11	3	6
3	15	5	0	16	12	5	8
4	2	5	11	0	9	13	14
5	8	6	3	5	0	6	7
6	6	13	8	11	5	0	5
7	5	15	3	7	12	6	0

After city 1, city 5 in both P_1 and P_2 is the un-visited city, city 5 is added that produces the offspring as (1, 5). After city 5, cities 3 in P_1 and 7 in P_2 are un-visited cities with costs c_{53}=3 and c_{57}=7. Since c_{57}>c_{53}, city 7 is added that produces the offspring as (1, 5, 7). After city 7, cities 4 in P_1 and 3 in P_2 are un-visited cities with costs c_{74}=7 and c_{73}=3. Since c_{74}>c_{73}, city 4 is added that produces the offspring as (1, 5, 7, 4). After city 4, city 6 in P_1 with costs c_{46}=13, but no city in P_2. So, for P_2, search from the beginning and finds un-visited city 3 with c_{43}=11. Since c_{46}>c_{43}, city 6 is added that produces the offspring as (1, 5, 7, 4, 6). After city 6, no city is present in P_1 and un-visited city 2 is present in P_2 with cost c_{62}=13. So, for P_1 search from the beginning and finds un-visited city 3 with c_{63}=8. Since c_{62}>c_{63}, city 2 is added that produces the offspring as (1, 5, 7, 4, 6, 2). Finally, after city 2, the only remaining city is 3, which is added that produces the offspring as (1, 5, 7, 4, 6, 2, 3) with cost 7 is obtained. Fig. 2 shows parents (P1 and P2) and offspring (O) chromosomes. In general, the crossover operator which preserves better characteristics of parents in their children is expected to be better, and SCX is expected to be better in this regard. In Fig. 2(c), bold five edges are from either parent.

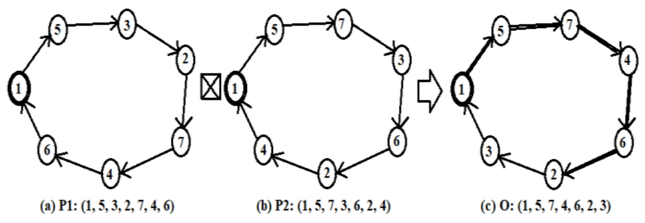
(a) P1: (1, 5, 3, 2, 7, 4, 6)
(b) P2: (1, 5, 7, 3, 6, 2, 4)
(c) O: (1, 5, 7, 4, 6, 2, 3)

Fig. 2. Result of SCX Operation for the MSTSP.

Though SCX is observed as the best method, however, sometimes it creates bad offspring. So, to maintain a mixture of offspring and parent in a population, we replace the 1st parent by the offspring if it is better. In addition, the 2-opt local search is used on the better offspring to improve it further. Since the SCX operator produces only an offspring. So, to keep population size same in all generations, when selecting next pair for crossover, the present 2nd chromosome will be selected as the 1st parent and the 3rd chromosome will be as the 2nd parent, and so forth.

E. Mutation Operator

As some weaker chromosomes are omitted in selection and crossover processes in any generation, so, there might be some stronger chromosomes' structures which were lost forever. So, normally, mutation is applied to regain them. In traditional mutation operations, a gene (position) is chosen arbitrarily in a chromosome, then alters its subsequent allele (city). Some of the mutation operators are inversion mutation, insertion mutation, swap mutation, adaptive mutation [36]. The adaptive mutation is implemented for our GAs. To perform this mutation, the information from the chromosomes in a population are gathered to identify a structure amongst them. If mutation is to be performed, then the chromosomes which do not match the structure would be muted. The steps of adaptive mutation are as follows:

Step 1: In the current population, take all chromosomes.

Step 2: Construct a one-dimensional array of order n, let A, by adding a city which appears least time in the present position of all chromosomes.

Step 3: If the mutation is allowed, two genes are selected arbitrarily so that they are not same in the subsequent positions of the array, A, then they are exchanged.

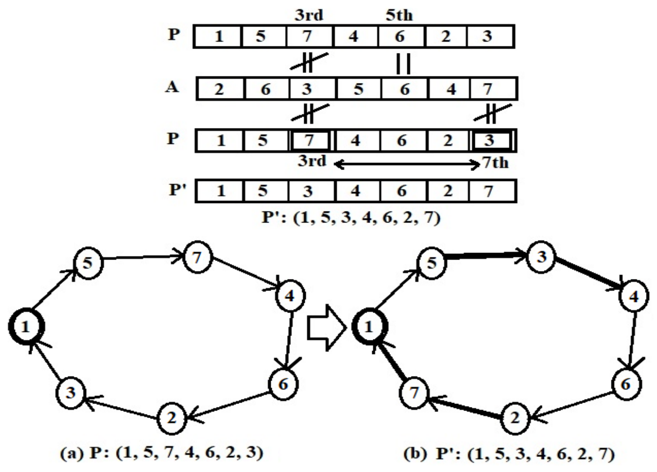
3rd
5th
P
1 5 7 4 6 2 3
A
2 6 3 5 6 4 7
P
1 5 7 4 6 2 3
3rd
7th
P'
1 5 3 4 6 2 7
P': (1, 5, 3, 4, 6, 2, 7)
(a) P: (1, 5, 7, 4, 6, 2, 3)
(b) P': (1, 5, 3, 4, 6, 2, 7)

Fig. 3. Result of Adaptive Mutation Operation for the MSTSP.

Since the city 1 is always fixed in the 1st position, we exclude the 1st position as well as the city 1 in this procedure. For example, suppose the chromosome P: (1, 5, 7, 4, 6, 2, 3) is chosen for mutation operation, and the array be A: [2, 6, 3, 5, 6, 4, 7]. Let 3rd and 5th positions are chosen arbitrarily. The 3rd position's gene '7' is same as the subsequent element in A, but 5th position's gene '6' is same as the subsequent element in A. So, we choose another position arbitrarily and let, 7th position is chosen that allows the swap. Hence, the mutated chromosome would be P': (1, 5, **3**, 4, 6, 2, **7**) that is showed in Fig. 3. New edges in the muted chromosome are shown in boldfaces in Fig. 3(b).

F. Local Search

There are various local search procedures presented in the literature, amongst them combined mutation is seen as a nice local search procedure ([2], [27], [28]). It merges insertion, inversion, and swap mutations with 1.00 probabilities. Insertion mutation selects a city in a chromosome, then inserts into an arbitrary position. Inversion mutation selects two positions in a chromosome, then inverts the sub-chromosome between them. Swap mutation selects two cities (genes) arbitrarily and exchanges them. We define these three mutations as local search procedures in our HGA as follows.

1) Insertion search: Suppose (α_1, α_2, α_3, ..., α_n) be a chromosome. The insertion search may be defined as:

Step 0: For i: = 2 to n-1 perform the next step.

Step 1: For j: = i+1 to n perform the next step.

Step 2: If inserting city α_i after city α_j improves the assignment cost, then insert the city α_i after the city α_j.

2) Inversion search: Suppose (α_1, α_2, α_3, ..., α_n) be a chromosome. The inversion search may be defined as:

Step 0: For i: = 2 to n-1 perform the next step.

Step 1: For j: = i+1 to n perform the next step.

Step 2: If inverting sub-chromosome between the cities α_i and α_j improves the assignment cost, then invert the sub-chromosome.

3) Swap search: Suppose (α_1, α_2, α_3, ..., α_n) be a chromosome. The swap search may be defined as:

Step 0: For i: = 2 to n-1 perform the next step.

Step 1: For j: = i+1 to n perform the next step.

Step 2: If exchanging cities α_i and α_j improves the assignment cost, then swap them.

In our local search procedure, one of these three local search is selected for our HGA for the problem.

G. Perturbation Procedure

Though GAs are very good methods, but sometimes, they get stuck in local optima. This may be due to identical population, and so, the population must be varied. Perturbation procedure is useful in escaping from local optima. If (Best Solution – Average Solution) < 0.10*Best Solution, then we apply partially mapped crossover (PMX), swap mutation and combined mutation operators. The PMX selects two crossover points, describes swap mappings in the segment between these points, and delivers two offspring. Further, mutation can assist other operators to beat local optima issue and thus, can find better solutions.

H. Hybrid GAs

In our study, a simple genetic algorithm (SGA) and four hybrid genetic algorithms (HGAs) are proposed for the MSTSP. The SGA starts with initial population generated by sequential sampling approach which is further improved by 2-opt search, and it is tried to improve gradually the population through stochastic remainder selection, sequential constructive crossover, and adaptive mutation. A stopping condition of maximum generation is adopted. The hybrid genetic algorithms (HGAs) include a selected local search and perturbation procedure to the proposed SGA. When the stopping condition is satisfied, near-optimal solution is produced. The selected local search defines each proposed HGA as follows:

HGA1: Insertion search,

HGA2: Inversion search,

HGA3: Swap search, and

HGA4: Randomly selected one of three local searches – insertion, inversion, and swap search.

The common structure of our proposed HGAs is presented in Fig. 4.

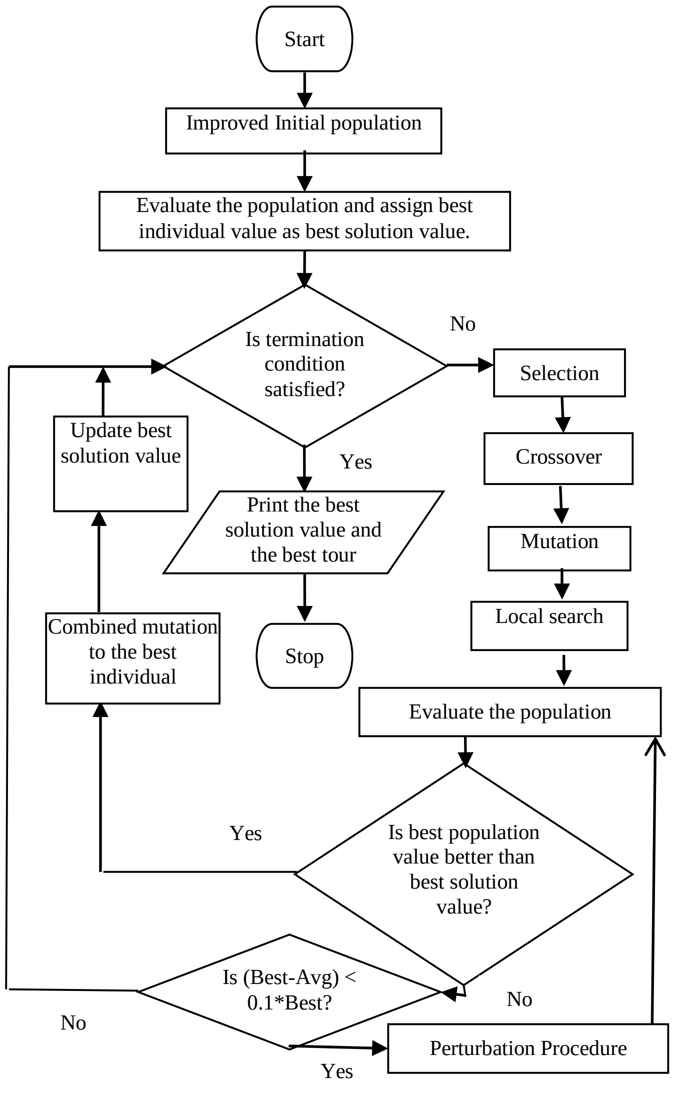
Start
Improved Initial population
Evaluate the population and assign best individual value as best solution value.
Is termination condition satisfied?
No
Selection
Crossover
Mutation
Local search
Evaluate the population
Yes
Print the best solution value and the best tour
Stop
Update best solution value
Combined mutation to the best individual
Is best population value better than best solution value?
Yes
No
Is (Best-Avg) < 0.1*Best?
No
Yes
Perturbation Procedure

Fig. 4. Flow-Chart of our Proposed HGAs.

IV. Computational Experience

We encoded our proposed SGA and HGAs in Visual C++. To determine the value of HGAs, computational experience is performed on some typical TSPLIB instances [37] of many sizes and then implemented on a Laptop with i7-1065G7 CPU@1.30 GHz and 8 GB RAM under MS Windows 10. We run GAs for separate parameter settings, and chosen parameters are recorded in Table II.

TABLE II. Comparative Study of SGA and HGAs for Asymmetric TSPLIB Instances

Instance	n	Result	SGA	HGA1	HGA2	HGA3	HGA4
ftv33	34	Best Sol	142	**143**	**143**	**143**	**143**
		Avg. Sol	134.45	**143.00**	**143.00**	**143.00**	**143.00**
		S.D.	4.25	0.00	0.00	0.00	0.00
		Avg. Time	0.08	0.16	0.16	0.08	0.14
		Avg. Imp(%)		6.36	6.36	6.36	6.36
ftv35	36	Best Sol	151	**154**	**154**	**154**	**154**
		Avg. Sol	141.50	**154.00**	**154.00**	**154.00**	**154.00**
		S.D.	5.35	0.00	0.00	0.00	0.00
		Avg. Time	0.39	0.78	0.86	0.71	0.78
		Avg. Imp(%)		8.83	8.83	8.83	8.83
ftv38	39	Best Sol	151	**154**	**154**	**154**	**154**
		Avg. Sol	140.00	**152.20**	151.00	151.20	152.00
		S.D.	6.42	2.65	3.32	3.43	2.80
		Avg. Time	0.90	1.40	1.18	1.13	1.02
		Avg. Imp(%)		8.71	7.86	8.00	8.57
p43	43	Best Sol	16	**17**	**17**	**17**	**17**
		Avg. Sol	13.20	**17.00**	**17.00**	**17.00**	**17.00**
		S.D.	0.30	0.00	0.00	0.00	0.00
		Avg. Time	0.98	1.32	1.30	1.17	1.04
		Avg. Imp(%)		28.79	28.79	28.79	28.79
ftv44	45	Best Sol	156	**162**	**162**	**162**	**162**
		Avg. Sol	145.25	161.20	161.50	**161.60**	**161.60**
		S.D.	7.10	1.60	1.50	1.20	1.20
		Avg. Time	0.94	1.55	1.43	1.57	1.60
		Avg. Imp(%)		10.98	11.19	11.26	11.26
ft47	48	Best Sol	162	**168**	**168**	**168**	**168**
		Avg. Sol	150.15	167.70	167.80	**167.90**	167.80
		S.D.	6.40	0.46	0.40	0.30	0.40
		Avg. Time	1.02	1.77	1.86	1.93	1.87
		Avg. Imp(%)		11.69	11.75	11.82	11.75
ry48p	48	Best Sol	1176	**1232**	**1232**	**1232**	1227
		Avg. Sol	1140.00	1211.40	1215.60	**1217.40**	1217.20
		S.D.	22.49	17.77	14.61	15.49	16.83
		Avg. Time	1.07	1.42	1.90	1.99	1.97
		Avg. Imp(%)		6.26	6.63	6.79	6.77
ft53	53	Best Sol	360	**379**	**379**	**379**	**379**
		Avg. Sol	345.70	376.00	**378.00**	376.50	376.50
		S.D.	12.52	2.45	2.00	2.50	2.50
		Avg. Time	1.52	1.74	2.02	1.97	1.99
		Avg. Imp(%)		8.76	9.34	8.91	8.91
ftv55	56	Best Sol	143	**154**	**154**	**154**	**154**
		Avg. Sol	132.60	151.60	152.80	153.60	**153.90**
		S.D.	8.01	3.75	2.14	0.92	0.30
		Avg. Time	1.73	2.02	2.18	2.09	2.19
		Avg. Imp(%)		14.33	15.23	15.84	16.06
ftv64	65	Best Sol	143	**160**	158	**160**	**160**
		Avg. Sol	132.30	154.20	153.60	153.60	**158.00**
		S.D.	6.08	3.19	3.07	5.39	2.00
		Avg. Time	1.78	2.70	2.54	2.52	2.54
		Avg. Imp(%)		16.55	16.10	16.10	19.43
ft70	70	Best Sol	926	973	972	**974**	**974**
		Avg. Sol	893.70	964.50	964.90	**967.40**	965.60
		S.D.	19.02	6.38	7.05	8.46	10.58
		Avg. Time	1.93	2.89	2.91	3.04	3.08
		Avg. Imp(%)		7.92	7.97	8.25	8.05
ftv70	71	Best Sol	147	160	161	160	**161**
		Avg. Sol	133.50	156.00	157.00	**157.60**	**157.60**
		S.D.	10.25	2.90	2.83	1.96	2.42
		Avg. Time	1.81	2.82	2.93	2.97	3.15
		Avg. Imp(%)		16.85	17.60	18.05	18.05
kro124p	100	Best Sol	2224	**2347**	**2347**	**2347**	**2347**
		Avg. Sol	2153.40	2345.10	**2347.00**	**2347.00**	**2347.00**
		S.D.	65.66	5.70	0.00	0.00	0.00
		Avg. Time	2.16	3.42	3.40	3.58	3.65
		Avg. Imp(%)		8.90	8.99	8.99	8.99
ftv170	171	Best Sol	146	172	170	172	**174**
		Avg. Sol	128.20	169.20	167.60	170.67	**171.67**
		S.D.	7.67	2.77	3.65	2.31	2.23
		Avg. Time	5.10	7.52	7.91	7.57	7.95
		Avg. Imp(%)		31.98	30.73	33.13	32.35
rbg323	323	Best Sol	14	**20**	19	**20**	**20**
		Avg. Sol	12.20	18.36	17.92	**18.73**	18.56
		S.D.	0.60	1.12	0.60	1.01	1.26
		Avg. Time	8.30	15.23	13.22	14.75	15.73
		Avg. Imp(%)		50.49	46.89	53.52	47.54
rbg358	358	Best Sol	16	**18**	**18**	**18**	**18**
		Avg. Sol	13.50	17.20	17.60	17.60	**17.71**
		S.D.	1.32	1.10	0.89	0.89	0.76
		Avg. Time	9.57	19.21	17.56	18.78	18.43
		Avg. Imp(%)		27.41	30.37	30.37	31.19
rbg403	403	Best Sol	15	15	16	**16**	**16**
		Avg. Sol	13.40	15.00	15.20	**15.29**	**15.29**
		S.D.	1.20	0.00	0.45	0.49	0.49
		Avg. Time	12.49	22.33	23.26	25.03	25.23
		Avg. Imp(%)		11.94	13.43	14.10	14.10

Fig. 5 shows solutions for ftv170 (only 100 generations are considered) by SGA and HGAs. Each curvature is for only one GA that shows progress of the solution in consecutive generations. The figure indicates some good variations of HGA4 and proves that HGA4 is the top. HGA3 also has certain variations that are placed in 2^{nd} place. But SGA has no variations at all after few generations, get stuck in local maximum so rapidly and is proven to be the worst.

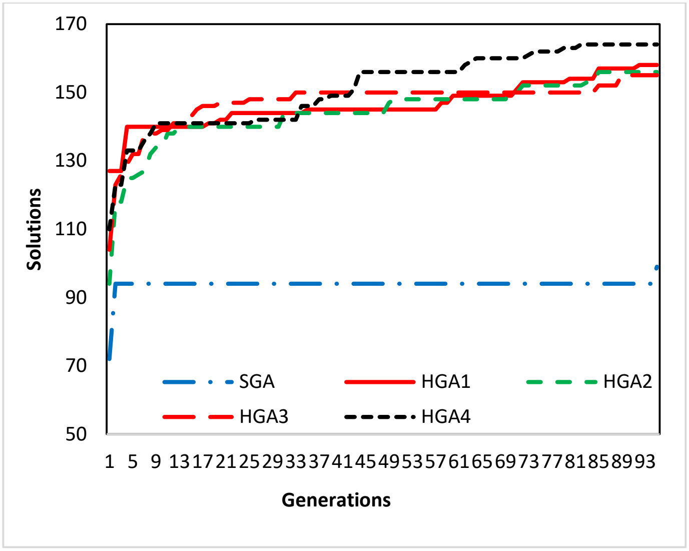

Fig. 5. Solutions by SGA and HGAs for ftv170.

Comparative studies among SGA and HGAs on some TSPLIB asymmetric and symmetric instances are reported in Tables III and V, respectively. We record best solution (Best Sol), average solution (Avg. Sol), standard deviation (S.D) of the solutions, and average computational time (Avg. Time) (in seconds) for each problem instance in the tables. The best results are indicated by boldfaces. The tables further report average improvement (%) of the average solution by HGAs over the average solution by SGA using the following formula:

$$\text{Avg. Imp}(\%) = 100(S_1 - S_2)/S_2 ,$$

where S_1 and S_2 are average solutions by a HGA and the SGA, respectively.

The Table III summarizes the results of asymmetric instances of sizes from 34 to 403. From the table, it is observed that the SGA could not find either best solution or best average solution for any instance. All four HGAs together obtained best average solutions with least S.D. for three instances ftv33, ftv35 and p43. In addition, HGA2, HGA3 and HGA4 together obtained best average solutions with lowermost S.D. for the instance kro124p; HGA3 and HGA4 together obtained best average solutions for ftv44, ftv70 and rbg403; HGA1 obtained best average solution with least S.D. for the instance ftv38; HGA2 obtained best average solution with lowest S.D. for the instance ft53; HGA3 obtained best average solutions for ftv47, ry48p, ft70 and rbg323; HGA4 obtained best average solutions with least S.D. for the instances ftv55, ftv64, ftv170 and rbg358. From this experiment we can say that HGA3 and HGA4 are competing, however, HGA4 is observed as the best algorithm.

By looking at the average improvement (%) of the average solutions by HGAs, we have the same conclusion. The average improvements (%) are shown in Fig. 6 that also indicates the suitability of the HGAs, specially HGA3 and HGA4. Looking at the overall results on the asymmetric instances, one can decide that the HGA4 is the best one and HGA3 is the 2nd best one.

TABLE III. PARAMETERS FOR ALL GAS

Parameters	**Values**
Population size	50
Crossover probability	100%
Mutation probability	9%
Termination criterion	For SGA, 2000 generations For HGAs, 200 generations
No. of runs for each instance	20 times

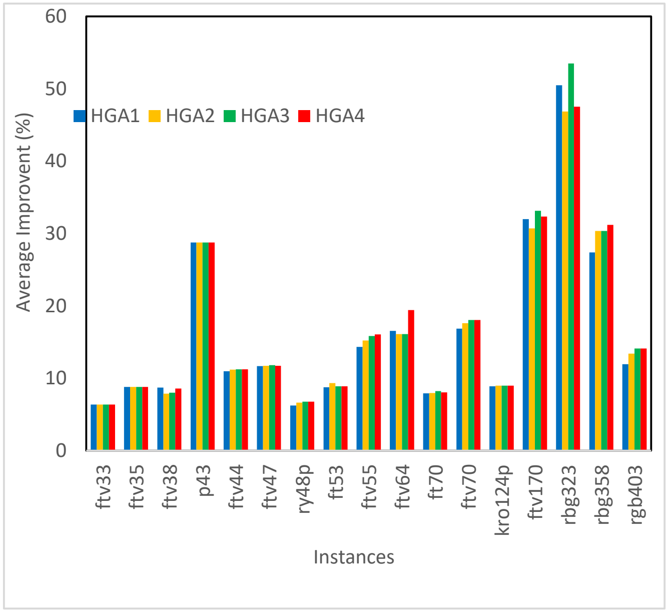

Fig. 6. Average Improvement (%) of Solutions by different HGAs over SGA for Asymmetric Instances.

It is very apparent from the above experiments that HGAs have very good improvements in the solutions over SGA for the TSPLIB asymmetric instances. HGA4 is observed as the best algorithm and SGA is the worst. Now, to verify whether average solutions obtained by HGA4 is significantly and statistically different from the average solutions obtained by remaining HGAs, Student's t-test is conducted using following formula [38]. It may be noted that 20 runs have been conducted for each instance.

$$t = \frac{\bar{X}_1 - \bar{X}_2}{\sqrt{\frac{SD_1^2}{n_1 - 1} + \frac{SD_2^2}{n_2 - 1}}}$$

$where,$

$\bar{X}_1 - average\ of\ first\ sample,$

$SD_1 - standard\ deviation\ of\ first\ sample,$

$\bar{X}_2 - average\ of\ second\ sample,$

$SD_2 - standard\ deviation\ of\ second\ sample,$

$n_1 - first\ sample\ size,$

$n_2 - second\ sample\ size,$

Here, $\bar{X}_2$ and SD_2 values are obtained by HGA4, and $\bar{X}_1$ and SD_1 values are obtained by other HGAs.

The t-statistic results are provided in Table IV. The t-values may be negative or positive. As the MSTSP is a maximization problem, negative value indicates that HGA4 obtained better solution than its competitive HGA, and positive value indicates that the competitive HGA obtained better solution than HGA4. Here 95% confidence level ($t_{0.05}$ = 1.73) is applied, so, if t-value is bigger than 1.73, their difference is significant. In this condition if t-value is negative then HGA4 is better, else competitive HGA is better. If t-value is smaller than 1.73, then they have no statistical and significant difference. The table further reports the name of the better HGA.

TABLE IV. THE T-VALUES AGAINST HGA4 AND THE RESULT ABOUT HGAS THAT OBTAINED SIGNIFICANTLY BETTER SOLUTIONS FOR THE TSPLIB ASYMMETRIC INSTANCES

Instance	HGA1	HGA2	HGA3	Instance	HGA1	HGA2	HGA3
ftv33	---	---	---	ftv64	-7.06	-8.41	-5.36
Better	---	---	---	Better	**HGA4**	**HGA4**	**HGA4**
ftv35	---	---	---	ft70	-0.62	-0.39	0.93
Better	---	---	---	Better	---	---	---
ftv38	0.36	-1.61	-1.26	ftv70	-2.97	-1.13	0.00
Better	---	---	---	Better	**HGA4**	---	---
p43	---	---	---	kro124p	-2.33	---	---
Better	---	---	---	Better	**HGA4**	---	---
Ftv44	-1.40	-0.36	0.00	ftv170	-4.86	-6.66	-2.18
Better	---	---	---	Better	**HGA4**	**HGA4**	**HGA4**
ftv47	-1.15	0.00	1.40	rbg323	-0.83	-3.21	0.74
Better	---	---	---	Better	---	**HGA4**	---
ry48p	-1.66	-0.50	0.06	rbg358	-2.67	-0.66	-0.66
Better	---	---	---	Better	**HGA4**	---	---
ft53	-1.00	3.28	0.00	rbg403	-4.14	-0.95	0.00
Better	---	**HGA2**	---	Better	**HGA4**	---	---
ftv55	-4.28	-3.56	-2.17				
Better	**HGA4**	**HGA4**	**HGA4**				

On ten instances, HGA4 and HGA1 have no significant and statistical difference. On the other seven instances HGA4 is better than HGA1. On twelve instances, HGA4 and HGA2 have no significant and statistical difference. On the instance ft53, HGA2 is better than HGA4, and on the remaining four instances HGA4 is better than HGA2. On fourteen instances, HGA4 and HGA3 have no significant and statistical difference, and on the remaining three instances, HGA4 is better than HGA3. On all seventeen instances, HGA4 is found better than other HGAs. From this experiment we can say that HGA4 is the best for asymmetric instances.

The Table V summarizes the results of symmetric instances of sizes from 21 to 318. From the table, it is observed that the SGA could obtain best solution for only the instance gr21. All HGAs obtained best average solutions with lowest S.D. for four instances gr21, fri26, bayg29 and berlin52. In addition, HGA2 and HGA4 together obtained best average solutions with lowest S.D. for the instances kroA150 and a280; HGA3 and HGA4 together obtained best average solutions with lowest S.D. for the instance si175; HGA2 obtained best average solutions for three instances - gr48, st70 and pr76; HGA3 obtained best average solutions for the instance dantzig42; HGA4 obtained best average solutions for five instances - eil51, lin105, ch130, d198, pr226 and lin318. From this study we can say that HGA4 is the best one.

TABLE V. COMPARATIVE STUDY OF SGA AND HGAS FOR SYMMETRIC TSPLIB INSTANCES

Instance	n	Result	SGA	HGA1	HGA2	HGA3	HGA4
gr21	21	Best Sol	**370**	**370**	**370**	**370**	**370**
		Avg. Sol	368.75	**370.00**	**370.00**	**370.00**	**370.00**
		S.D.	4.44	0.00	0.00	0.00	0.00
		Avg. Time	0.00	0.01	0.00	0.01	0.00
		Avg. Imp(%)		0.34	0.34	0.34	0.34
fri26	26	Best Sol	100	**102**	**102**	**102**	**102**
		Avg. Sol	93.80	**102.00**	**102.00**	**102.00**	**102.00**
		S.D.	2.60	0.00	0.00	0.00	0.00
		Avg. Time	0.05	0.06	0.05	0.06	0.05
		Avg. Imp(%)		8.74	8.74	8.74	8.74
bayg29	39	Best Sol	182	**189**	**189**	**189**	**189**
		Avg. Sol	167.40	**189.00**	**189.00**	**189.00**	**189.00**
		S.D.	9.14	0.00	0.00	0.00	0.00
		Avg. Time	0.06	0.08	0.07	0.08	0.07
		Avg. Imp(%)		12.90	12.90	12.90	12.90
dantzig42	42	Best Sol	69	**73**	**73**	**73**	**73**
		Avg. Sol	63.15	71.30	71.30	**73.00**	72.80
		S.D.	2.57	0.71	0.71	0.00	0.71
		Avg. Time	0.90	1.28	1.25	1.15	1.07
		Avg. Imp(%)		12.91	12.91	15.60	15.28
gr48	48	Best Sol	515	558	**559**	558	558
		Avg. Sol	486.00	553.00	**558.00**	554.70	557.90
		S.D.	13.54	7.27	0.45	6.36	0.30
		Avg. Time	1.06	1.56	1.58	1.61	1.47
		Avg. Imp(%)		13.79	14.81	14.14	14.79
eil51	51	Best Sol	33	38	38	38	**39**
		Avg. Sol	30.15	37.20	37.70	37.70	**39.00**
		S.D.	1.77	0.84	0.64	0.64	0.00
		Avg. Time	1.13	1.56	1.49	1.90	1.55
		Avg. Imp(%)		23.38	25.04	25.04	29.35
berlin52	52	Best Sol	504	**541**	**541**	**541**	**541**
		Avg. Sol	466.55	**541.00**	**541.00**	**541.00**	**541.00**
		S.D.	16.90	0.00	0.00	0.00	0.00
		Avg. Time	1.54	1.77	1.84	1.67	1.80
		Avg. Imp(%)		15.96	15.96	15.96	15.96
st70	70	Best Sol	57	**63**	**63**	**63**	**63**
		Avg. Sol	52.00	59.55	**62.48**	60.10	62.25
		S.D.	2.88	1.56	1.43	1.30	1.22
		Avg. Time	1.88	2.02	2.10	2.04	2.09
		Avg. Imp(%)		14.52	16.31	15.58	15.87

pr76	76	Best Sol	8698	**9214**	**9214**	**9214**	**9214**
		Avg. Sol	7852.20	9085.60	**9169.90**	9043.65	9126.15
		S.D.	633.96	156.30	80.83	185.41	93.28
		Avg. Time	1.99	2.76	2.85	2.59	2.62
		Avg. Imp(%)		15.71	16.78	15.17	16.22
lin105	105	Best Sol	1270	1460	**1474**	**1474**	**1474**
		Avg. Sol	1173.85	1455.15	1466.35	1452.40	**1469.65**
		S.D.	83.07	8.76	5.72	12.46	8.16
		Avg. Time	2.14	3.65	3.78	3.89	3.98
		Avg. Imp(%)		23.54	24.92	23.73	25.20
ch130	130	Best Sol	395	454	455	457	**458**
		Avg. Sol	355.30	450.80	448.30	452.55	**458.00**
		S.D.	20.58	6.96	7.09	5.02	0.00
		Avg. Time	2.61	4.18	3.94	4.07	4.20
		Avg. Imp(%)		26.88	26.18	27.37	28.91
kroA150	150	Best Sol	1818	2147	**2153**	**2153**	**2153**
		Avg. Sol	1733.65	2132.50	**2153.00**	2137.29	**2153.00**
		S.D.	65.91	10.02	0.00	21.02	0.00
		Avg. Time	2.91	4.43	4.51	4.83	4.89
		Avg. Imp(%)		23.01	24.19	23.28	24.19
si175	175	Best Sol	276	296	**304**	**304**	**304**
		Avg. Sol	243.52	287.35	285.56	**304.00**	**304.00**
		S.D.	29.51	10.25	12.63	0.00	0.00
		Avg. Time	5.18	8.21	8.12	8.78	8.06
		Avg. Imp(%)		9.79	17.26	24.84	24.84
d198	198	Best Sol	643	731	735	731	**738**
		Avg. Sol	571.80	721.50	729.10	717.20	**738.00**
		S.D.	28.81	10.07	9.84	14.41	0.00
		Avg. Time	5.58	8.79	8.63	8.07	8.77
		Avg. Imp(%)		26.18	27.51	25.43	29.07
pr226	226	Best Sol	8070	9301	9357	9353	**9360**
		Avg. Sol	7811.40	9201.30	9272.90	9256.10	**9360.00**
		S.D.	72.12	104.91	15.94	58.74	0.00
		Avg. Time	6.23	10.73	10.83	9.91	10.75
		Avg. Imp(%)		17.79	18.71	18.49	19.82
a280	280	Best Sol	101	145	**148**	145	**148**
		Avg. Sol	93.62	144.40	**148.00**	135.30	**148.00**
		S.D.	7.85	4.03	0.00	2.00	0.00
		Avg. Time	7.85	13.02	13.09	13.03	13.10
		Avg. Imp(%)		54.24	58.09	44.52	58.09
lin318	318	Best Sol	1870	2351	**2395**	2375	**2395**
		Avg. Sol	1654.55	2351.50	2387.20	2361.30	**2388.10**
		S.D.	137.45	17.66	10.48	32.33	10.25
		Avg. Time	9.10	16.95	15.23	15.42	15.24
		Avg. Imp(%)		42.12	44.28	42.72	44.34

By looking at the average improvement (%) of the average solutions by HGAs, we can have the same conclusion. These results are shown in Fig. 7 that also shows the usefulness of the HGAs, specially HGA2 and HGA4. Looking at the overall results on the symmetric instances, one can conclude that the HGA4 is the best one and HGA2 is the second best one.

From the experiment we can say that HGAs have very good improvements in the solution over SGA for the TSPLIB symmetric instances. HGA4 is found to be the best and SGA is the worst. Now, to verify whether average solutions obtained by HGA4 is significantly and statistically different from the average solutions obtained by other HGAs, Student's t-test is conducted, and the results are provided in Table VI.

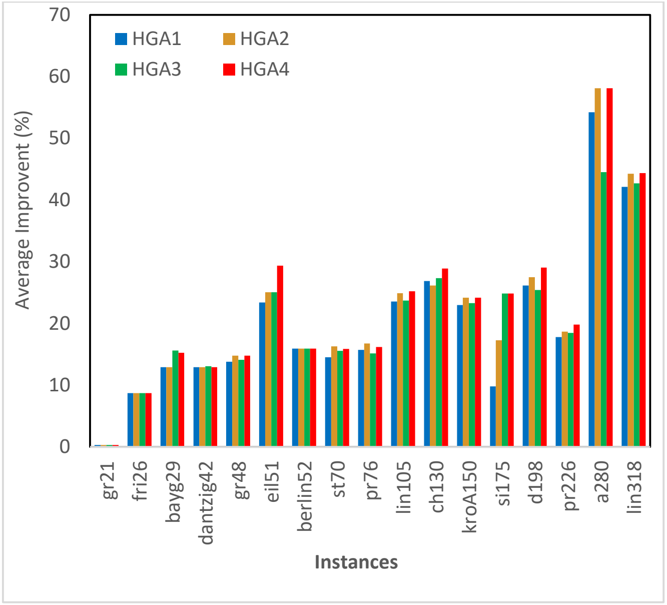

Fig. 7. Average Improvement (%) of Solutions by different HGAs over SGA for Symmetric Instances.

TABLE VI. THE T-VALUES AGAINST HGA4 AND THE RESULT ABOUT HGAS THAT OBTAINED SIGNIFICANTLY BETTER SOLUTIONS FOR THE TSPLIB SYMMETRIC INSTANCES

Instance	**HGA1**	**HGA2**	**HGA3**	**Instance**	**HGA1**	**HGA2**	**HGA3**
gr21	---	---	---	lin105	-11.40	-2.32	-8.11
Better	---	---	---	Better	**HGA4**	**HGA4**	**HGA4**
fri21	---	---	---	ch130	-7.24	-9.58	-7.60
Better	---	---	---	Better	**HGA4**	**HGA4**	**HGA4**
bayg29	---	---	---	kroA150	-14.32	---	-5.23
Better	---	---	---	Better	**HGA4**	---	**HGA4**
dantzig42	-10.46	-10.46	1.31	si175	-11.37	-10.22	---
Better	**HGA4**	**HGA4**	---	Better	**HGA4**	**HGA4**	---
gr48	-4.71	1.29	-3.52	d198	-11.47	-6.33	-10.10
Better	**HGA4**	---	**HGA4**	Better	**HGA4**	**HGA4**	**HGA4**
eil51	-15.00	-14.22	-14.22	pr226	-21.73	-14.62	-16.83
Better	**HGA4**	**HGA4**	**HGA4**	Better	**HGA4**	**HGA4**	**HGA4**
berlin52	---	---	---	a280	-6.25	---	-44.45
Better	---	---	---	Better	**HGA4**	---	**HGA4**
st70	-2.47	0.86	-0.59	lin318	-11.60	-0.37	-5.37
Better	**HGA4**	---	---	Better	**HGA4**	---	**HGA4**
pr76	-1.56	2.48	-2.78				
Better	---	**HGA2**	**HGA4**				

Looking at the Table VI, on five instances, HGA4 and HGA1 have no statistical and significant differences, and on the other twelve instances HGA4 is better than HGA1. On nine instances there is no statistically significant difference between HGA4 and HGA2; on the instance pr76, HGA2 is better than HGA4; and on the remaining seven instances, HGA4 is better than HGA2. On seven instances there is no statistically significant difference between HGA4 and HGA3, and on remaining ten instances, HGA4 is better than HGA3. On all seventeen instances, HGA4 is found better than other HGAs.

From this experiment, we can say that HGA4 is the best algorithm for symmetric TSPLIB instances also. Hence, for all asymmetric and symmetric instances, HGA4 is the best algorithm. To decide the second best algorithm for both categories of instances, we performed Student's t-test between HGA2 and HGA3 and reported in Table VII. From the table, it is found that out of thirty four instances, on nineteen instances, HGA2 and HGA3 have no statistical and significant differences. On the other nine instances HGA2 is better than HGA3, and on six instances, HGA3 is better than HGA2. Hence, HGA2 is the second best and HGA3 is the third best.

We now compare our proposed HGA4 with a state-of-art algorithm, namely, multi-start iterated local search (MS-ILS(h_1+h_2)) [4] on some TSPLIB symmetric instances of sizes from 21 to 318. We record best solution (BS), worst solution (WS), average solution (AS), and computational time (Time) (in seconds) for each problem instance in Table VIII. Better solutions are shown in boldfaces.

Looking at the average solutions, for the four instances, namely, dantzig42, gr48, lin105 and lin318 our HGA4 could find better solutions than solutions found by MS-ILS(h_1+h_2). For another two instances, namely, st70 and pr226, solutions by MS-ILS(h_1+h_2) are better. For the remaining instances, solutions are same. Of course, MS-ILS(h_1+h_2) takes less computational time. Overall, looking at the solution quality, our suggested HGA4 is found to be better.

TABLE VII. THE T-VALUES OF HGA2 AGAINST HGA3 AND THE RESULT ABOUT HGAS THAT OBTAINED SIGNIFICANTLY BETTER SOLUTIONS FOR THE TSPLIB ASYMMETRIC AND SYMMETRIC INSTANCES

Instance	**HGA2**	**Instance**	**HGA2**	**Instance**	**HGA2**	**Instance**	**HGA2**
ftv33	---	ftv35	---	ftv38	-0.29	p43	---
Better	---	Better	---	Better	----	Better	---
ftv44	-0.36	ftv47	-1.40	ry48p	-0.59	ft53	3.28
Better	---	Better	---	Better	---	Better	**HGA2**
ftv55	-2.40	ftv64	0.00	ft70	-1.59	ftv70	-1.22
Better	**HGA3**	Better	---	Better	---	Better	---
kro124p	---	ftv170	-4.98	rbg323	-4.83	rbg358	0.00
Better	---	Better	**HGA3**	Better	**HGA3**	Better	---
rbg403	-0.95	gr21	---	fri21	---	bayg29	---
Better	---	Better	---	Better	---	Better	---
dantzig42	-11.13	gr48	3.62	eil51	0.00	berlin52	---
Better	**HGA3**	Better	**HGA2**	Better	---	Better	---
st70	1.38	pr76	4.37	lin105	7.12	ch130	-3.42
Better	---	Better	**HGA2**	Better	**HGA2**	Better	**HGA3**
kroA150	5.23	si175	-10.22	d198	4.77	pr226	4.37
Better	**HGA2**	Better	**HGA3**	Better	**HGA2**	Better	**HGA2**
a280	44.45	lin318	5.33				
Better	**HGA2**	Better	**HGA2**				

TABLE VIII. THE T-VALUES AGAINST HGA4 AND THE RESULT ABOUT HGAS THAT OBTAINED SIGNIFICANTLY BETTER SOLUTIONS FOR THE TSPLIB SYMMETRIC INSTANCES

Instance	**n**	**MS-ILS(h_1+h_2)**					**HGA4**			
		BS	**WS**	**AS**	**Time**		**BS**	**WS**	**AS**	**Time**
gr21	21	370	370	370.00	0.06		370	370	370.00	0.00
fri26	26	102	102	102.00	0.09		102	102	102.00	0.05
bayg29	29	189	189	189.00	0.11		189	189	189.00	0.07
dantzig42	42	73	71	72.60	0.21		73	72	**72.80**	1.07
gr48	48	558	545	555.40	0.29		558	557	**557.90**	1.47
eil51	51	39	39	39.00	0.31		39	39	39.00	1.55
berlin52	52	541	541	541.00	0.32		541	541	541.00	1.80
st70	70	63	63	**63.00**	0.54		63	62	62.25	2.09
pr76	76	9214	9214	**9214.00**	0.70		9214	9069	9126.15	2. 62
lin105	105	1474	1460	1467.50	1.28		1474	1462	**1469.65**	3.98
ch130	130	458	458	458.00	1.99		458	458	458.00	4.20
kroA150	150	2153	2153	2153.00	2.37		2153	2153	2153.00	4.89
si175	175	304	304	304.00	3.05		304	304	304.00	8.54
d198	198	738	738	738.00	3.92		738	738	738.00	8.77
pr226	226	9360	9360	9360.0	5.05		9360	9350	9360.00	10.75
a280	280	148	148	148.00	7.36		148	148	148.00	13.10
lin318	318	2387	2381	2385.70	12.45		2395	2385	**2388.10**	15.24

V. Conclusion and Discussion

In this paper, a simple GA (SGA) and four hybrid GAs (HGA1, HGA2, HGA3 and HGA4) have been proposed for solving the MSTSP. The SGA used initial population by a sequential sampling, a proportionate selection, sequential constructive crossover, and adaptive mutation. Three local search procedures based on inversion, insertion and swap mutations, and a perturbation procedure have been used in different HGAs. The usefulness of the HGAs have been examined amongst themselves and calculated percentage of improvement of the obtained solution over the solution by SGA for the asymmetric and symmetric TSPLIB problem instances. The results show a very good improvement of the solutions by HGAs over the solutions by SGA. Further, it is seen that for asymmetric instances, HGA3 is placed in 2nd position and HGA4 is the best one. For symmetric instances, HGA2 is placed in 2nd position and HGA4 is the best one. Overall, for both categories of the instances, HGA4 is the best one, HGA2 is the second best and HGA3 is the 3rd best. Further, a comparative study is carried out between HGA4 and by multi-start iterated local search (MS-ILS(h1+h2)). Looking at the solution quality, our suggested HGA4 is found to be better.

Though our proposed HGAs obtained very efficient solutions with slight differences between best solutions and average solutions, however, we admit that still there is an opportunity to improve the solutions by combining better local search procedures, another heuristic method and perturbation procedure to the instances that is under our study.

Acknowledgment

This research was supported by Deanery of Academic Research, Imam Mohammad Ibn Saud Islamic University, Saudi Arabia vide Grant No. 18-11-09-010. The first author thanks the Deanery for its financial support. The authors further thank the honorable anonymous reviewers for their constructive comments and suggestions which helped the authors to improve this paper.

References

[1] E.M. Arkin, Y.-J. Chiang, J.S.B. Mitchell, S.S. Skiena, and T.-C. Yang, On the maximum scatter traveling salesperson problem, SIAM Journal of Computing 29 (1999) 515–544.

[2] Z.H. Ahmed, A hybrid genetic algorithm for the bottleneck traveling salesman problem. ACM Transactions on Embedded Computing Systems 12 (2013) Art. No. 9.

[3] A. Barvinok, S.P. Fekete, D.S. Johnson, A. Tamir, G.J. Woeginger and R. Woodroofe, The geometric maximum traveling salesman problem, Journal of the ACM 50(5) (2003) 641–664.

[4] P. Venkatesh, A. Singh and R. Mallipeddi, A multi-start iterated local search algorithm for the maximum scatter traveling salesman problem, 2019 IEEE Congress on Evolutionary Computation (CEC), Wellington, New Zealand, 2019, pp. 1390-1397.

[5] J. LaRusic and A.P. Punnen, the asymmetric bottleneck traveling salesman problem: Algorithms, complexity and empirical analysis, Computers & Operations Research 43 (2014) 20–35.

[6] F. Scholz, Coordination hole tolerance stacking, Technical Report BCSTECH-93-048, Boeing Computer Services, November 1993.

[7] L.R. John, the bottleneck traveling salesman problem and some variants, Master of Science of Simon Fraser University, Canada, 2010.

[8] W.B. Carlton and J.W. Barnes, Solving the travelling salesman problem with time windows using tabu search, IEE Transaction 28 (1996) 617–629.

[9] J.W. Ohlmann and B.W. Thomas, A compressed-annealing heuristic for the traveling salesman problem with time windows, INFORMS Journal of Computing 19 (1) (2007) 80–90.

[10] C.-B. Cheng and C.-P. Mao, A modified ant colony system for solving the travelling salesman problem with time windows, Mathematical Computer Modelling 46 (2007) 1225–1235.

[11] M. Gendreau, A. Hertz, G. Laporte and M. Stan, A generalized insertion heuristic for the traveling salesman problem with time windows, Operations Research 46 (3) (1998) 330–335.

[12] R.F. da Silva and S. Urrutia, A general VNS heuristic for the traveling salesman problem with time windows, Discrete Optimization 7 (4) (2010) 203–211.

[13] A. S. Hameed, M. L. Mutar, H. M. B. Alrikabi, Z. H. Ahmed, Abdul–A. A. Razaq, & H. K. Nasser. A Hybrid Method Integrating a Discrete Differential Evolution Algorithm with Tabu Search Algorithm for the Quadratic Assignment Problem: A New Approach for Locating Hospital Departments. Mathematical Problems in Engineering, 2021.

[14] DE. Goldberg. Genetic algorithms in search, optimization, and machine learning, Addison-Wesley, New York, 1989.

[15] Z.H. Ahmed, Genetic algorithm for the traveling salesman problem using sequential constructive crossover operator, International Journal of Biometrics & Bioinformatics 3 (2010) 96-105.

[16] Z.H. Ahmed, A comparative study of eight crossover operators for the maximum scatter travelling salesman problem, International Journal of Advanced Computer Science and Applications (IJACSA) 11 (2020) 317-329.

[17] Z.H. Ahmed, Algorithms for the quadratic assignment problem, LAP LAMBERT Academic Publishing, Latvia, Mauritius, 2019.

[18] A.S. Hameed, B.M. Aboobaider, N.H. Choon, M.L Mutar, W.H. Bilal. 'Review on the Methods to Solve Combinatorial Optimization Problems Particularly: Quadratic Assignment Model', International Journal of Engineering & Technology, 7, pp. 15–20. 2018.

[19] M.L. Mutar, M.A. Burhanuddin, A.S. Hameed, N. Yusof, H.J. Mutashar. 'An efficient improvement of ant colony system algorithm for handling capacity vehicle routing problem', International Journal of Industrial Engineering Computations, 11(4), pp. 549–564. 2020. DOI: 10.5267/j.ijiec.2020.4.006.

[20] Yi-J. Chiang. New approximation results for the maximum scatter TSP. Algorithmica, 41 (2005) 309–341.

[21] S.N. Kabadi and A.P. Punnen. The bottleneck TSP, In the Traveling Salesman Problem and Its Variations, G. Gutin and A.P. Punnen (eds.), Chapter 15, Kluwer Academic, Dordrecht, 2002.

[22] I. Hoffmann, S. Kurz, and J. Rambau, The maximum scatter TSP on a regular grid, in Operations Research Proceedings 2015, Springer, 2017, pp. 63–70.

[23] W. Dong, X. Dong and Y. Wang, The improved genetic algorithms for multiple maximum scatter traveling salesperson problems, In J. Li et al. (Eds.): CWSN 2017, CCIS 812, pp. 155–164, 2018.

[24] Z.H. Ahmed, A lexisearch algorithm for the bottleneck travelling salesman problem, International Journal of Computer Science and Security 3(5) (2010) 569-577.

[25] Z.H. Ahmed, A data-guided lexisearch algorithm for the bottleneck travelling salesman problem, International Journal of Operational Research 12(1) (2011) 20-33.

[26] Z.H. Ahmed, A hybrid sequential constructive sampling algorithm for the bottleneck traveling salesman problem, International Journal of Computational Intelligence Research 6(3) (2010) 475-484.

[27] Z.H. Ahmed, A hybrid genetic algorithm for the bottleneck traveling salesman problem, ACM Transactions on Embedded Computing Systems (TECS)12(1) (2013) 1-10.

[28] Z.H. Ahmed, An experimental study of a hybrid genetic algorithm for the maximum traveling salesman problem, Mathematical Sciences 7(1) (2013) 1-7.

[29] K. Deb, Optimization for engineering design: algorithms and examples, Prentice Hall of India Pvt. Ltd., New Delhi, India, 1995.

[30] L. Davis, Job-shop scheduling with genetic algorithms, Proceedings of an International Conference on Genetic Algorithms and Their Applications, 136-140, 1985.

[31] D.E. Goldberg and R. Lingle, Alleles, loci and the travelling salesman problem, In J.J. Grefenstette (ed.) Proceedings of the 1st International Conference on Genetic Algorithms and Their Applications. Lawrence Erlbaum Associates, Hilladale, NJ, 1985.

[32] I.M. Oliver, D. J. Smith and J.R.C. Holland, A Study of permutation crossover operators on the travelling salesman problem, In J.J. Grefenstette (ed.). Genetic Algorithms and Their Applications: Proceedings of the 2nd International Conference on Genetic Algorithms. Lawrence Erlbaum Associates, Hilladale, NJ, 1987.

[33] J. Grefenstette, R. Gopal, B. Rosmaita and D. Gucht, Genetic algorithms for the traveling salesman problem, In Proceedings of the First International Conference on Genetic Algorithms and Their Applications, (J. J. Grefenstette, Ed.), Lawrence Erlbaum Associates, Mahwah NJ, 160–168, 1985.

[34] N.J. Radcliffe and P.D. Surry, Formae and variance of fitness, In D. Whitley and M. Vose (Eds.) Foundations of Genetic Algorithms 3, Morgan Kaufmann, San Mateo, CA, 51-72, 1995.

[35] D. Whitley, T. Starkweather and D. Shaner, the traveling salesman and sequence scheduling: quality solutions using genetic edge recombination, In L. Davis (Ed.) Handbook of Genetic Algorithms. Van Nostrand Reinhold, New York, 350-372, 1991.

[36] Z.H. Ahmed, an improved genetic algorithm using adaptive mutation operator for the quadratic assignment problem, 38th International Conference on Telecommunications and Signal Processing 2015 (TSP 2015) (2015) 1-5.

[37] G. Reinelt, TSPLIB, http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/

[38] M. Nikolić and D. Teodorović, “Empirical study of the bee colony optimization (BCO) algorithm,” Expert Systems with Applications, vol. 40, pp. 4609–4620, 2013.