Optimization of the Travelling Salesman Problem Using a New Hybrid Genetic Algorithm

Zakir Hussain Ahmed [1,*], Furat Fahad Altukhaim[2], Abdul Khader Jilani Saudagar[3], and Shakir Khan[4]

[1]Department of Mathematics and Statistics, College of Science, Imam Mohammad Ibn Saud Islamic University (IMSIU), Riyadh 11432, Saudi Arabia.
[2]Department of Computer Science, Science Departments at Al Quwaiiyah, Shaqra University, Shaqra, Saudi Arabia.
[3]Information Systems Department, Imam Mohammad Ibn Saud Islamic University (IMSIU), Riyadh 11432, Saudi Arabia
[4]College of Computer and Information Sciences, Imam Mohammad Ibn Saud Islamic University (IMSIU), Riyadh 11432, Saudi Arabia. [*]Correspondence: *zaahmed@imamu.edu.sa*

Abstract
The travelling salesman problem is very famous and very difficult combinatorial optimization problem that has several applications in operations research, computer science and industrial engineering. As the problem is difficult, finding its optimal solution is computationally very difficult. Thus, several researchers have developed heuristic/metaheuristic algorithms for finding heuristic solutions to the problem instances. In this present study, a new hybrid genetic algorithm (HGA) is suggested to find heuristic solution to the problem. In our HGA we used comprehensive sequential constructive crossover, adaptive mutation, 2-opt search and a new local search algorithm along with a replacement method, then executed our HGA on some standard TSPLIB problem instances, and finally, we compared our HGA with simple genetic algorithm and an existing state-of-the-art method. The experimental studies show the effectiveness of our proposed HGA for the problem.

Keywords:
Travelling salesman problem; Hybrid genetic algorithm; Comprehensive sequential constructive crossover; Adaptive mutation; Local search; Replacement method.

1. Introduction

The travelling salesman problem (TSP) is very famous and very difficult NP-hard [1] combinatorial optimization problem (COP). It was first recognized by Euler in 1759 as the Knights' tour problem. The problem is defined as: There is a network consisting of n nodes (or cities) containing the headquarters at 'node 1'. For each node-pair (i, j), a distance (or travel time or travel cost) matrix, $D=[d_{ij}]$, is provided. The problem aims to find a minimum distance (or minimum cost or minimum time) Hamiltonian cycle (Figure 1) [2]. Asymmetric TSP and symmetric TSP are two major types of the TSP. If $d_{ij} = d_{ji}$, for every node pair (i, j), then the TSP is symmetric otherwise, the TSP is asymmetric.

The problem has numerous theoretical and practical applications, such as automatic drilling of PCBs and circuits [3], scheduling [4], etc. This problem can be solved by exact and heuristic algorithms. Using basic exact methods, one has to explore lot of possible solutions to find an optimal solution. As the possible number of solutions of the problem is very large in each TSP type, it is very tough problem. Furthermore, since the TSP can be applied to model numerous other complicated problems, several researchers suggested several exact and approximation approaches for obtaining solution of the problem.

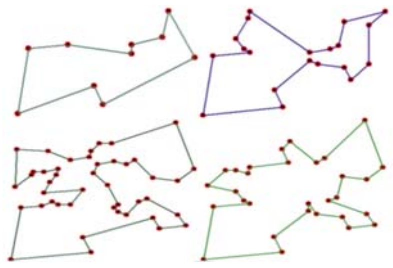
Fig. 1. Examples of sub-optimum solutions for different TSPs.

Branch-and-bound [5], integer programming [6], branch-and-cut [7], lexisearch [8] are some exact algorithms. However, problem instances of size more than 100 can never be solved easily optimally by any exact approach in a sufficient time, and there are numerous real-life problem instances of sizes more than 100 are available. To solve such problem instances, we have to use heuristic/metaheuristic algorithms. Generally, heuristic algorithms cannot guarantee that the best obtained solution is always optimal. Of course, they can obtain solution which is very close to the optimal solution very quickly. However, sometimes they obtain a constant solution in each iteration of the algorithm. So, in that case, it is beneficial to execute the algorithm using different starting points to find a better solution, because they may not escape from the local optimums (Figure 2) and so, get stuck in these local optimums [9].

Manuscript received March 5, 2024
Manuscript revised March 20, 2024
https://doi.org/10.22937/IJCSNS.2024.24.3.2

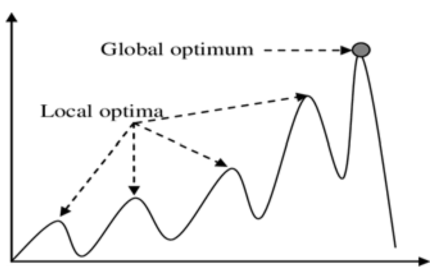

Fig. 2. Local and global optimums in a complex optimization problem

One of important and advanced versions of heuristic algorithms are metaheuristic algorithms which have resemblances with natural or social systems. They are found to be applied successfully to solve several COPs. Guided by clients or parameters, they can find very good solutions within an acceptable computational time, and generally, they do not get stuck in local optimums. Of course, the parameters setting is not an easy task and arbitrary parameter setting plays an important role. Examples of metaheuristic algorithms for solving the TSP are genetic algorithm [10], ant colony system [11], tabu search [12], simulated annealing [13], particle swarm optimization [14], etc. out of these algorithms, genetic algorithm is supposed to be better algorithm for the TSP and for the other COPs.

Genetic algorithm (GA) originated from the evolutionary process of biology. It is supposed to be an efficient and robust metaheuristic algorithm for solving large-scale COPs. However, simple GA cannot find optimal solutions for large-scale problem instances. Hence, many researchers combined metaheuristic methods to propose hybrid algorithms to obtain better solutions within very short time ([15], [16]). In this paper, we are developing a hybrid genetic algorithm (HGA) for the TSP. First, we plan to develop simple genetic algorithm (SGA) using comprehensive sequential constructive crossover [17] and adaptive mutation [18] for the problem. Then, 2-opt search, a local search and a replacement method are incorporated into the SGA for developing a HGA to obtain better solutions. Effectiveness of the proposed HGA is examined against ant colony system algorithm [2] to the problem for some standard TSPLIB instances.

This paper is structured as follows: Section 2 reports a brief review on the algorithms for the TSP. A hybrid genetic algorithm is presented in Section 3 for the TSP. Section 4 presents computational experience for our proposed algorithm. Finally, discussion and concluding remarks are presented in Section 5.

2. Literature Review

Several exact and heuristic algorithms have been developed for solving the TSP. Regarding the exact algorithm, in [19], the TSP was solved by formulating it as integer programming and then found an optimal solution to a 42-node problem instance. In [20], a lexisearch algorithm is developed for obtaining optimal solution to the asymmetric TSP. It is reported that the algorithm takes diverse computational times for numerous instances of same size. In [8], a data-guided lexisearch algorithm is developed to solve the asymmetric TSP. In [21], a branch and bound approach based on the assignment relaxation was developed to solve the TSP instances. In [7], a branch-and-cut algorithm was developed for solving large-scale symmetric TSP instances.

Regarding the heuristic algorithm, in [22], an evolutionary approach was developed to find solution for the TSP. In [11], ant colony optimization algorithm was proposed to solve the problem. In [12], a tabu search algorithm was proposed for the undirected selective TSP. In [14], a particle swarm optimization algorithm was proposed to solve the problem. In [9], a simple genetic algorithm utilizing sequential constructive crossover operator was developed for obtaining solution to the TSP. In [13], a list-based simulated annealing algorithm was suggested for the problem. In [23], a heuristic algorithm based on the minimum spanning tree is developed for the problem. In [24[, a discrete spider monkey optimization algorithm, based on the behavior of monkeys, was proposed to solve the problem, and compared the algorithm with other algorithms. In [25], a memetic algorithm using 3-opt method was developed to find better solution to the problem, and then compared its results with some state-of-the-art algorithms. In [26], a discrete grey wolf optimizer algorithm, based on the behavior of grey wolves, was suggested to solve the problem. Further, a 2-opt approach was combined to the algorithm to improve the algorithm and then compared with some other algorithms. In [27], a Harris hawk optimization algorithm using the Lin and Kernighan heuristic, a local search method and Metropolis acceptance rule was developed to solve the problem, and then solved some benchmark problem instances. In [28], a colony optimization algorithm was proposed by combining searching ants and then compared successfully with the traditional algorithm. Recently, in [2], a meta-based ant colony system algorithm was proposed to find solution to the problem. The algorithm does not apply local pheromone update but applies global pheromone update. The algorithm is compared with other algorithms and found effective results.

3. A Hybrid Genetic Algorithm for the TSP

GA is one of the best heuristic algorithms to solve the TSP. It originated from the evolutionary process of biology that imitates the theory of survival of the fittest [10]. The simple GA starts from a chromosome set, known as initial population, and then applies three operator- selection, crossover and mutation, to obtain heuristic solution to the given problem. Hybrid GA incorporates local search or other heuristic methods to the simple GA.

3.1 Initial Population and Selection Operator

The TSP solutions are characterized by chromosomes that are permutation of given nodes. For example, the tour {1→6→8→7→3→10→2→4→9→ 5 →1} corresponds to (1, 6, 8, 7, 3, 10, 2, 4, 9, 5) for a 10-node problem instance. The total distance (or cost) of the tour defines its objective function, and its multiplicative inverse defines the fitness function. An initial population of P_s size is generated at random. To generate a mating pool, the stochastic remainder selection procedure is applied.

3.2 Crossover Operator

In crossover operation, pair of chromosomes mate to produce offspring chromosome(s). However, the traditional crossover operation may not be able to produce valid offspring chromosomes for the TSP. For the TSP, several crossover rules are presented in the literature. Some of them are distance-based and remaining are blind crossover operators. Distance-based crossover operators use distance between nodes and blind crossover operators neither use distances or the problem data to create offspring chromosome(s). Some common distance-based crossover operators are greedy crossover [29], heuristic crossover [30], distance-preserving crossover [31], sequential constructive crossover (SCX) [32], etc. Some popular blind crossover operators are partially mapped crossover [33], ordered crossover [29], cycle crossover [34], etc. In [17], a comprehensive SCX (CSCX) was proposed by combining greedy SCX (GSCX) (Algorithm 1) and reverse GSCX (RGSCX) (Algorithm 2) and compared CSCX with five other crossover operators. Comparative studies showed the effectiveness of the CSCX. We consider this CSCSX with crossover probability, P_c, for our SGA and HGA. Here, GSCX and RGSCX produce first and second offsprings respectively.

Algorithm 1: *Greedy sequential constructive crossover algorithm*

```
Input: D, $P_c$, Pair of parent chromosomes.
Output: Offspring chromosome.
Generate random number $r \in [0,1]$.
if ($r = P_c$) then do
    Set $p = 1$.
    The offspring chromosome contains only 'node 1'.
    for $i = 2$ to $n$ do
        In each parent chromosome consider the first un-visited' node found after 'node p'.
        if no 'un-visited' node is found in a parent, then
            Consider the closest 'un-visited' node from the group of remaining 'unvisited' nodes and concatenate it to the partially constructed offspring chromosome'. Rename the present node as 'node p' and continue to the next iteration.
        end if
        Suppose 'node $\alpha$' and 'node $\beta$' are selected in 1st and 2nd chromosomes respectively.
        if ($d_{\alpha p} < d_{\beta p}$) then do
            Add 'node $\alpha$' to the offspring chromosome.
        else
            Add 'node $\beta$' to the offspring chromosome.
        end if
        Rename the present node as 'node p' and continue.
    end for
end if
Return the offspring chromosome
```

Let P1: (1, 7, 9, 3, 2, 4, 8, 5, 10, 6) and P2: (1, 6, 3, 9, 4, 5, 7, 8, 2, 10) be parent chromosomes having the headquarters at ‘node 1’. With respect to the distance matrix reported in Table 1, total distances of P1 and P2 are 558 and 447 respectively. Using GSCX on P1 and P2, first offspring O1: (1, 6, 4, 5, 7, 9, 3, 8, 10, 2) with distance 328 can be obtained. The obtained chromosome is further improved by using 2-opt search method.

Node	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	999	46	54	16	53	12	60	29	54	35
2	46	999	100	56	79	45	75	65	81	32
3	54	100	999	47	65	58	82	48	63	82
4	16	56	47	999	38	12	47	41	38	51
5	53	79	65	38	999	43	18	78	4	85
6	12	45	58	12	43	999	49	41	44	42
7	60	75	82	47	18	49	999	88	21	88
8	29	65	48	41	78	41	88	999	77	38
9	54	81	63	38	4	44	21	77	999	86
10	35	32	82	51	85	42	88	38	86	999

Table 1: The Distance matrix

Using RGSCX on the P1 and P2, second offspring O2: (1, 2, 10, 8, 3, 5, 7, 9, 4, 6, 1) with distance 330 can be obtained. The obtained chromosome is further improved by using 2-opt search method.

```
Algorithm 2: Reverse greedy sequential constructive crossover algorithm
    Input: D, $P_c$, Pair of parent chromosomes.
    Output: Offspring chromosome.
    Generate random number $r \in [0,1]$.
    if ($r = P_c$) then do
        Suppose the 'node α' and the 'node β' are the last nodes in 1st
        and 2nd parent respectively. Then for selecting the last node,
        we check whether $d_{\alpha 1} < d_{\beta 1}$. If yes, then select 'node α',
        otherwise, 'node β' as the last node of the partially constructed
        offspring chromosome. Then rename this present node as
        'node p'.
        for i = n-1 down to 2 do
            In each parent chromosome consider the first un-visited'
            node found before 'node p'.
            if no 'un-visited' node is found in a parent, then
                Consider the closest 'un-visited' node from the
                group of remaining 'un-visited' nodes and
                concatenate it to the partially constructed offspring
                chromosome'. Rename the present node as 'node p'
                and continue to the next iteration.
            end if
            Suppose 'node α' and 'node β' are selected in 1st and 2nd
            chromosomes respectively.
            if ($d_{\alpha p} < d_{\beta p}$) then do
                Add 'node α' to the offspring chromosome.
            else
                Add 'node β' to the offspring chromosome.
            end if
            Rename the present node as 'node p' and continue.
        end for
    end if
    Return the offspring chromosome
```

3.3 Mutation Operator

In mutation operation, a location in a chromosome is chosen and its value is replaced to produce muted chromosome. But this traditional mutation operation may not create valid chromosome for the TSP. For the TSP, some popular mutation operators are displacement mutation, exchange mutation, insertion mutation, inversion mutation, etc. [10]. In [17], an adaptive mutation was proposed for the quadratic assignment problem. We are going to apply this adaptive mutation (Algorithm 3) with mutation probability, P_m, for our SGA and HGA.

```
Algorithm 4: Local search algorithm
    Input: A chromosomes.
    Output: Improved chromosome.
    for $i = 1$ to $n-1$ do
        for $j = i+1$ to $n$ do
            If inserting gene $\alpha_i$ after gene $\alpha_j$ reduces the distance of
              the tour, then insert the gene $\alpha_i$ after the gene $\alpha_j$.
            end if
            If inverting substring between the genes $\alpha_i$ and $\alpha_j$ reduces
              the present tour distance, then invert the substring.
            end if
            If displacing substring between the genes $\alpha_i$ and $\alpha_j$ at a
              random place reduces the present tour distance, then
              displace the substring.
            end if
        end for
    end for
    Return the improved chromosome.
```

3.5 Replacement Method

To further improve our GA search, the population should be varied, and so, a replacement method is applied. First an offspring is created using multi-parent sequential constructive crossover [36], it is then improved using 2-opt search and finally inserted into the population if the population is identical. Our hybrid GA is reported in Algorithm 5.

```
Algorithm 5: Hybrid genetic algorithm
    Input: $n$, $P_s$, $P_c$, $P_m$.
    Output: Best chromosome with its value.
    Initialize a random population of size $P_s$.
    Evaluate the population.
    Generation = 0.
    While the stopping condition is not satisfied.
        Generation = Generation + 1.
        Select fitter chromosomes by selection operator.
        Perform CSCX operator with crossover probability $P_c$.
        Perform adaptive mutation with mutation probability $P_m$.
        Perform local search method.
        If the population is identical, use replacement method.
    end while
    Return the best chromosome with its value.
```

4. Computational experiments

Our simple GA (SGA) and hybrid GA (HGA) are encoded in Visual C++ and run on some TSPLIB instances [37] on a Laptop with Intel(R) Core(TM) i7-1065G7 CPU @ 1.30GHz and 8.00GB RAM under MS Windows 11. The following GA parameter values are used: $P_s = 100$, $P_c = 0.95$, $P_m = 0.15$, Maximum generation = 50 for HGA and Maximum generation = 1500 for SGA. We have performed the experiments 50 times for each instance. The solution quality is measured by the percentage of excess (EX(%)) of the solution (S) to the best known solution (BKS) shown in [37], by: EX(%) = 100×(S/BKS - 1).

We report the best solution (BS), average solution (AS), EX(%) of the best solution (BE(%)), and EX(%) of average solution (AE(%)) over the BKS in 50 runs, standard deviation (SD) of the obtained solutions in 50 runs, percentage of improvement of average solution by HGA over the average solution by SGA (AI(%)). Table 2 reports comparative study between SGA and HGA for 14 asymmetric TSPLIB problem instances over 50 runs.

Looking at Table 2, the HGA is observed better than the SGA. Looking at BE(%), the SGA could find the best known solution for only instance – ftv44, whereas HGA could find best known solution for all fourteen instances. Furthermore, looking at AE(%), the SGA could not find the best known solution on average, whereas HGA could find best known solution on average for only one instance – p43. Finally, looking at AI(%), average solution by HGA has achieved minimum 0.34% of improvement and maximum 11.15% of improvement over SGA. So, it is confirmed that HGA is the improvement of SGA. The percentage of average solution excess by both algorithms are also shown in Figure 3, which demonstrates the effectiveness of HGA.

Also, we compared the solutions obtained by SGA and HGA in each generation (up to 50 generations) for the ftv170 instance and it is shown in Figure 4. From this figure it is also shown that HGA is improved over SGA.

Table 2: Comparative study between SGA and HGA for 14 asymmetric TSPLIB instances.

Instance	n	BKS	SGA					HGA					
			BS	AS	SD	BE(%)	AE(%)	BS	AS	SD	BE(%)	AE(%)	AI(%)
ftv33	34	1286	1314	1365.14	11.33	2.18	6.15	1286	1289.28	23.12	0.00	0.26	5.88
ftv35	36	1473	1497	1518.20	10.88	1.63	3.07	1473	1478.46	5.42	0.00	0.37	2.69
ftv38	39	1530	1548	1585.54	12.51	1.18	3.63	1530	1532.54	4.32	0.00	0.17	3.46
p43	43	5620	5631	5639.04	4.29	0.20	0.34	5620	5620.00	0.00	0.00	0.00	0.34
ftv44	45	1613	1613	1632.64	20.01	0.00	1.22	1613	1615.24	1.54	0.00	0.14	1.08
ftv47	48	1776	1799	1867.22	33.80	1.30	5.14	1776	1793.73	5.22	0.00	1.00	4.10
ry48p	48	14422	14859	15090.70	120.96	3.03	4.64	14422	14550.13	6.23	0.00	0.89	3.72
ft53	53	6905	7529	7793.38	128.28	9.04	12.87	6905	7093.82	11.24	0.00	2.73	9.86
ftv55	56	1608	1620	1663.08	16.87	0.75	3.43	1608	1610.96	2.56	0.00	0.18	3.24
ftv64	65	1839	1862	1891.30	12.92	1.25	2.84	1839	1860.39	6.58	0.00	1.16	1.66
ft70	70	38673	40130	41034.50	457.55	3.77	6.11	38673	39552.01	25.38	0.00	2.27	3.75
ftv70	71	1950	1977	2008.82	16.13	1.38	3.02	1950	1973.46	39.27	0.00	1.20	1.79
kro124p	100	36230	39332	40286.04	458.39	8.56	11.20	36230	37116.79	132.54	0.00	2.45	8.54
ftv170	171	2755	3120	3216.04	46.47	13.25	16.73	2755	2893.32	59.87	0.00	5.02	11.15

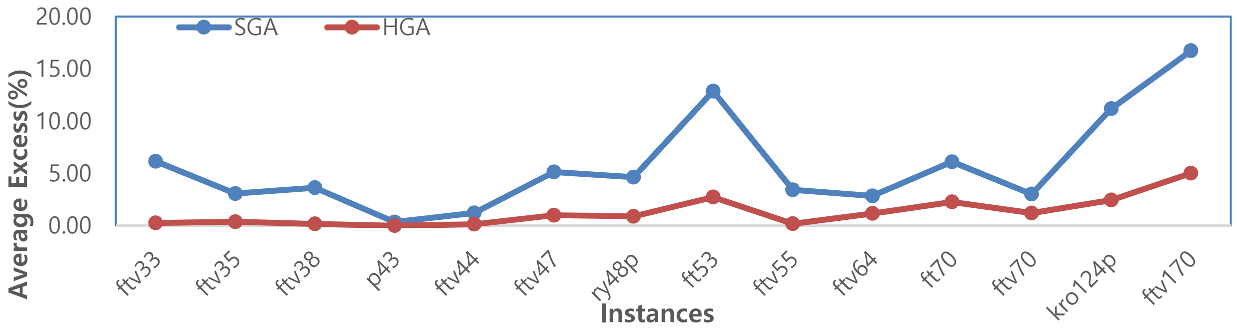

Fig. 3. Average Excess(%) by SGA and HGA for some asymmetric TSPLIB instances

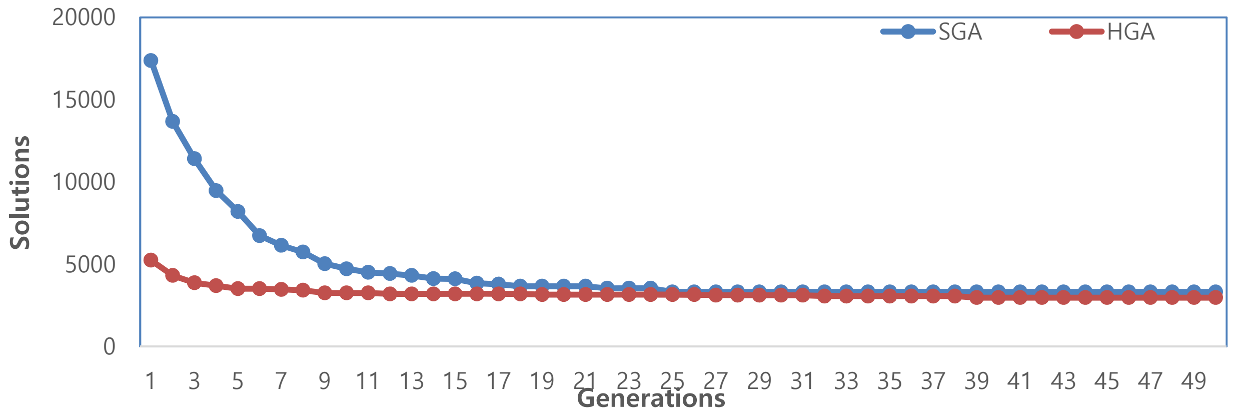

Fig. 4. Solutions obtained by SGA and HGA in each generation for the ftv170 instance

We further compare SGA and HGA for the symmetric TSPLIB instances. Table 3 reports comparative study between SGA and HGA for 14 symmetric TSPLIB problem instances over 50 runs.

Table 3: Comparative study between SGA and HGA for 14 symmetric TSPLIB instances.

Instance	n	BKS	SGA					HGA					
			BS	**AS**	**SD**	**BE(%)**	**AE(%)**	**BS**	**AS**	**SD**	**BE(%)**	**AE(%)**	**AI(%)**
gr21	21	2707	2707	2707.00	0.00	0.00	0.00	2707	2707.00	0.00	0.00	0.00	0.00
fri26	26	937	937	937.00	0.00	0.00	0.00	937	937.00	0.00	0.00	0.00	0.00
bayg29	29	1610	1610	1620.40	4.02	0.00	0.65	1610	1610.00	0.00	0.00	0.00	0.65
dantzig42	42	699	699	699.50	1.51	0.00	0.07	699	699.00	0.00	0.00	0.00	0.07
eil51	51	426	426	427.80	0.63	0.00	0.42	426	426.00	0.00	0.00	0.00	0.42
berlin52	52	7542	7542	7622.71	44.30	0.00	1.07	7542	7542.00	0.00	0.00	0.00	1.07
pr76	76	108159	112050	114277.83	797.23	3.60	5.66	108159	108187.02	89.33	0.00	0.03	5.63
lin105	105	14379	15493	15959.42	98.82	7.75	10.99	14379	14379.00	0.00	0.00	0.00	10.99
pr226	226	80369	89708	90430.06	446.69	11.62	12.52	80369	80920.23	185.48	0.00	0.69	11.75
gil262	262	2378	2612	2723.12	42.12	9.84	14.51	2378	2423.56	107.26	0.00	1.92	12.36
a280	280	2579	2884	2980.23	48.32	11.83	15.56	2579	2617.32	152.02	0.00	1.49	13.87
pr299	299	48191	54230	57120.08	917.27	12.53	18.53	48579	49060.45	198.47	0.81	1.80	16.43
lin318	318	42029	47738	49738.23	480.47	13.58	18.34	42415	42939.78	165.32	0.92	2.17	15.83
gr431	431	171414	195843	204807.66	3481.06	14.25	19.48	173132	175997.02	395.67	1.00	2.67	16.37

Looking at Table 3, the HGA is observed better than the SGA. Looking at BE(%), the SGA could find the best known solution for six instances – gr21, fri26, bayg29, dantzig42, eil51 and berlin52, whereas HGA could find best known solution for all fourteen instances. Furthermore, looking at AE(%), the SGA could find the best known solution on average for two instances, whereas HGA could find best known solution on average for seven instances. Finally, looking at AI(%), for two instances, SGA and HGA could find optimal and same solutions and for the remaining instances, average solution by HGA has achieved minimum 0.07% of improvement and maximum 16.43% of improvement over SGA. So, it is confirmed that HGA is the improvement of SGA for symmetric instances also. The percentage of average solution excess by both algorithms are also shown in Figure 5, which too demonstrates the effectiveness of our proposed HGA. By looking at the figure it is very clear that as the size of the problem instance increases the average improvement by the HGA also increases.

In [2],

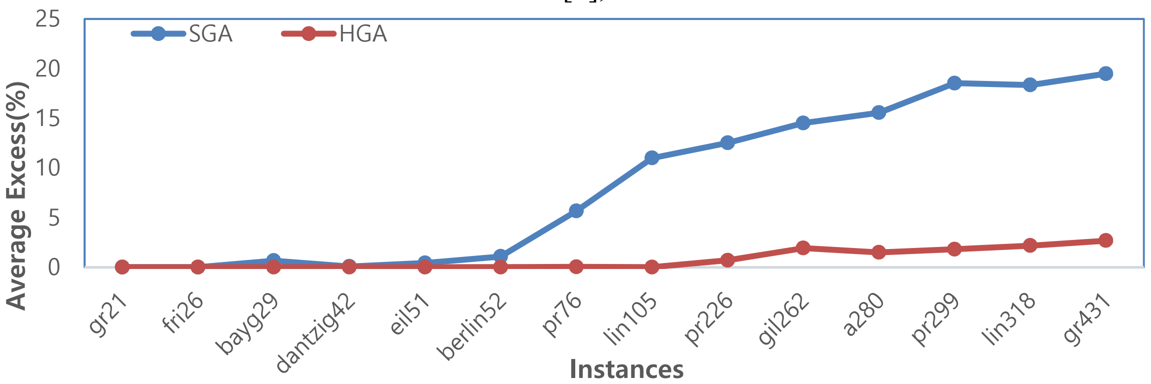

Fig. 5. Average Excess(%) by SGA and HGA for some symmetric TSPLIB instances

Manuscript received March 5, 2024
Manuscript revised March 20, 2024
https://doi.org/10.22937/IJCSNS.2024.24.3.2

In [2], a comparative study among eight different algorithms was presented on some symmetric TSPLIB instances and found that ant colony system algorithm (ACSA) is the best one. Now, we compare our HGA with the ACSA (Table 4).

Table 4: Comparative study between our HGA and ACSA for 14 symmetric TSPLIB instances.

Instance	**n**	**BKS**	**ACSA**				**HGA**				
			BS	**AS**	**BE(%)**	**AE(%)**	**BS**	**AS**	**BE(%)**	**AE(%)**	**AG(%)**
GR48	48	5046	5046	5046.00	0.00	0.00	5046	5046.00	0.00	0.00	0.00
ATT48	48	10628	10628	10628.00	0.00	0.00	10628	10628.00	0.00	0.00	0.00
Eil51	51	426	426	426.00	0.00	0.00	426	426.00	0.00	0.00	0.00
Berlin52	52	7542	7542	7542.00	0.00	0.00	7542	7542.00	0.00	0.00	0.00
ST70	70	675	675	682.00	0.00	1.04	675	675.00	0.00	0.00	1.04
Eil76	76	538	538	538.00	0.00	0.00	538	538.00	0.00	0.00	0.00
KroA100	100	21282	21282	21462.00	0.00	0.85	21282	21292.02	0.00	0.05	0.80
KroB100	100	22141	22141	22141.00	0.00	0.00	22141	22141.00	0.00	0.00	0.00
Eil101	101	629	629	636.00	0.00	1.11	629	633.36	0.00	0.69	0.42
Lin105	105	14379	14379	14384.00	0.00	0.03	14379	14379.00	0.00	0.00	0.03
KroA150	150	26524	26524	26726.00	0.00	0.76	26524	26613.92	0.00	0.34	0.42
KroB150	150	26130	26130	26352.00	0.00	0.85	26130	26307.06	0.00	0.68	0.17
KroA200	200	29368	29451	29883.00	0.28	1.75	29368	29761.78	0.00	1.34	0.41
KroB200	200	29437	29506	29802.00	0.23	1.24	29437	29561.32	0.00	0.42	0.81

Looking at Table 4, the HGA is observed better than the ACSA. Looking at BE(%), the ACSA could find the best known solution for twelve instances out of fourteen instances, whereas HGA could find best known solution for all fourteen instances. Furthermore, looking at AE(%), the ACSA could find the best known solution on average for six instances, whereas HGA could find best known solution on average for eight instances. Finally, looking at AG(%) (percentage of average solution gap between ACSA and HGA), HGA has achieved minimum 0.03% and maximum 1.04% of improved solution over ACSA. So, it is confirmed that HGA is better than ACSA. The percentage of average solution excess by both algorithms are also shown in Figure 6, which too demonstrates the effectiveness of HGA.

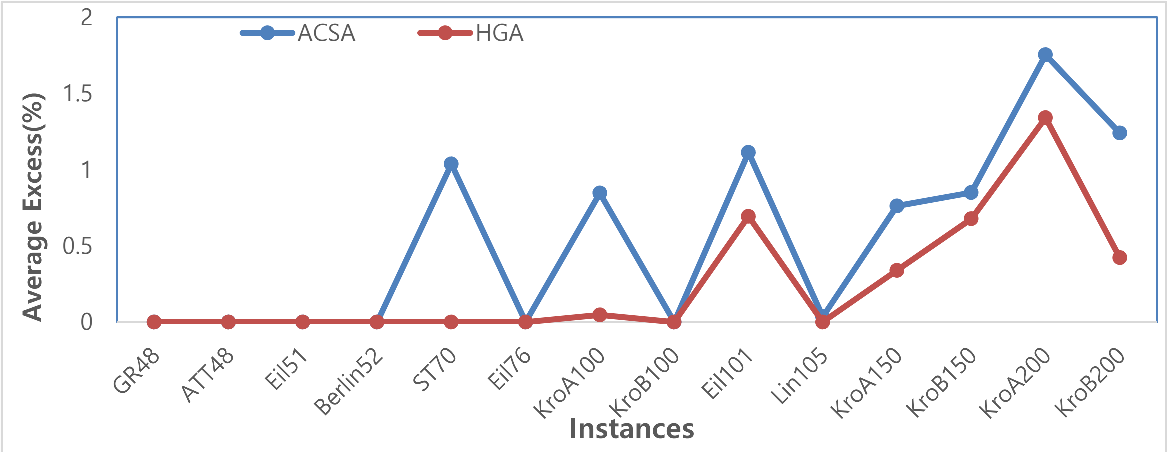

Fig. 6. Average Excess(%) by ACSA and HGA for some symmetric TSPLIB instances

5. Conclusion and Future Works

In this paper, a hybrid GA (HGA) is suggested to obtain heuristic solution for the TSP. In our proposed algorithm, comprehensive sequential constructive crossover, adaptive mutation, 2-opt search and a local search algorithm along with a replacement method were used. First, our HGA is compared with simple GA (SGA) for some asymmetric and symmetric TSPLIB problem instances. On both types of problem instances, our HGA has achieved very good improvement over SGA. The productivity of our HGA is very competent, and the algorithm has found high quality solutions to the selected problem instances of both asymmetric and symmetric types. Finally, we compared our HGA with an ant colony system algorithm (ACSA). Looking at the comparative study one can conclude that our suggested HGA is better than ACSA.

Though our proposed HGA could find best known solution at least once in 50 runs for the problem instances of size less than 300, however, for the problem instances of size more than 300, the proposed algorithm could not touch the best known solution. So, using a better initial population as well as a better local search algorithm or any other heuristic algorithm can provide better solutions, and even optimal solutions to the bigger sized problem instances. Finally, the proposed algorithm can be applied to other variants of the TSP, for example, time-dependent TSP, the general TSP, the TSP with backhauls, etc. The execution of these ideas would be postponed to our future articles. Further, we plan to study this problem using other metaheuristic algorithms, like football game algorithm [38], etc.

Conflicts of interests

All authors declare that they have no conflicts of financial interests regarding the publication of this paper.

Data Availability

The data used to support the findings of this study is available at the website: http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/tsp.

Acknowledgement

The authors extend their appreciation to the Deanship of Scientific Research, Imam Mohammad Ibn Saud Islamic University (IMSIU), Saudi Arabia, for funding this research work through Grant No. (221412020).

References

[1]. Arora S. (1998). Polynomial time approximation schemes for Euclidean traveling salesman and other geometric problems, *Journal of ACM*, 45(5), pp. 753–782.

[2]. Ahmed ZH, Yousefikhoshbakht M, Saudagar AK, and Khan S. (2023). Solving travelling salesman problem using an ant colony system algorithm, *International Journal of Computer Science and Network Security*, 23(2), pp. 55-64.

[3]. Ravikumar CP. (1992). Solving large-scale travelling salesperson problems on parallel machines, *Microprocessors and Microsystems*, 16(3), pp. 149-158.

[4]. Su F, Kong L, Wang H, and Wen Z. (2021). Modeling and application for rolling scheduling problem based on TSP, Applied Mathematics and Computation, 407, 126333.

[5]. Little JDC, Murthy KG, Sweeny DW, and Karel C. (1963). An algorithm for the travelling salesman problem, *Operations Research*, 11, pp. 972-989.

[6]. Sherali HD, Sarin SC, and Tsai Pei-F. (2006). A class of lifted path and flow-based formulations for the asymmetric traveling salesman problem with and without precedence constraints, *Discrete Optimization*, 3 , pp. 20-32.

[7]. Padberg M, and Rinaldi G. (1991). A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems, *SIAM Review*, 33(1), pp. 60–100.

[8]. Ahmed ZH. (2011). A data-guided lexisearch algorithm for the asymmetric traveling salesman problem, *Mathematical Problems in Engineering*, 2011, Article ID 750968, 18 pages.

[9]. Ahmed ZH. (2010). Genetic algorithm for the traveling salesman problem using sequential constructive crossover operator, *International Journal of Biometrics and Bioinformatics*, 6(3), 96-105.

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

[11]. Dorigo M, and Gambardella LM. (1997). Ant colonies for the traveling salesman problem, *Biosystems*, 43(2), pp. 73–81.

[12]. Gendreau M, Laporte G, and Semet F. (1998). A tabu search heuristic for the undirected selective travelling salesman problem, *European Journal of Operational Research*, 106(2–3), pp. 539-545.

[13]. Zhan S-H, Lin J, Zhang Z-J, and Zhong Y-W. (2016). List-based simulated annealing algorithm for traveling salesman problem, *Computational Intelligence and Neuroscience*, 2016, Article ID 1712630, 12 pages.

[14]. Wang K-P, Huang L, Zhou C-G, and Pang W. (2003). Particle swarm optimization for traveling salesman problem, *Proceedings of the 2003 International Conference on Machine Learning and Cybernetics (IEEE Cat. No.03EX693)*, Xi'an, 3, pp. 1583-1585.

[15]. Ahmed ZH. (2014). The ordered clustered travelling salesman problem: A hybrid genetic algorithm, *The Scientific World Journal*, 2014, Article ID 258207, 13 pages.

[16]. Ahmed ZH. (2018). A hybrid algorithm combining lexisearch and genetic algorithms for the quadratic assignment problem, *Cogent engineering*, 5(1), Article No. 1423743.

[17]. Ahmed ZH. (2020). Genetic algorithm with comprehensive sequential constructive crossover for the travelling salesman problem, *International Journal of Advanced Computer Science and Applications(IJACSA)*, 11(5), pp. 245-254.

[18]. Ahmed ZH. (2014). An improved genetic algorithm using adaptive mutation operator for the quadratic assignment problem, *Proceedings of 37th International Conference on Telecommunications and Signal Processing 2014 (TSP 2014)*, Berlin, Germany, pp. 616-620.

[19]. Dantzig GB, Fulkerson DR, and Johnson SM. (1954). Solution of a Large-scale Travelling Salesman Problem, *Operations Research*, 2, pp. 393-410.

[20]. Pandit SNN. (1964). An Intelligent approach to Travelling Salesman Problem, *Symposium in Operations Research*, Khragpur: Indian Institute of Technology.

[21]. Balas E, and Tooth P. (1992). Branch and Bound method. In: Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G., Shmoys, D.B. (Eds.). The Travelling Salesman Problem. New York: Wiley, pp. 361-402.

[22]. Fogel DB. (1988). An evolutionary approach to the travelling salesman problem, *Biological Cybernetics*, 60(2), pp. 139-144.

[23]. Kumar S, Munapo E, Lesaoana M, and Nyamugure P. (2018). A minimum spanning tree based heuristic for the travelling salesman tour, *Opsearch*, 55, pp. 150–164.

[24]. Akhand MAH, Ayon SI, Shahriyar SA, Siddique N, and Adeli H. (2019). Discrete spider monkey optimization for travelling salesman problem, *Applied Soft Computing*, 86, 105887.

[25]. Eremeev AV, and Kovalenko YV. (2020). A memetic algorithm with optimal recombination for the asymmetric travelling salesman problem, *Memetic Computing*, 12, pp. 23–36.

[26]. Panwar K, and Deep K. (2021). Discrete grey wolf optimizer for symmetric travelling salesman problem, *Applied Soft Computing*, 105, pp. 1–12.

[27]. Gharehchopogh FS, and Abdollahzadeh B. (2022). An efficient harris hawk optimization algorithm for solving the travelling salesman problem, *Cluster Computing*, 25, pp. 1981–2005.

[28]. Gao W. (2020). New ant colony optimization algorithm for travelling salesman problem, *International Journal of Computing and Intelligence System*, 13, pp. 44–55.

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

[30]. Grefenstette JJ. (1987). Incorporating problem specific knowledge into genetic algorithms, In L. Davis (Ed.), *Genetic algorithms and simulated annealing*, London, UK: Pitman / Pearson, pp. 42–60.

[31]. Freisleben B, and Merz P. (1996). A genetic local search algorithm for solving symmetric and asymmetric traveling salesman problems, in *Proceedings of the 1996 IEEE International Conference on Evolutionary Computation*, Nagoya, Japan, pp.616-621.

[32]. Ahmed ZH. (2014). Improved genetic algorithms for the traveling salesman problem, *International Journal of Process Management and Benchmarking*, 4(1), pp. 109-124.

[33]. Goldberg DE, and Lingle R. (1985). 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.

[34]. Oliver IM, Smith DJ, and Holland JRC. (1987). 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.

[35]. Ahmed ZH (2010). A hybrid sequential constructive sampling algorithm for the bottleneck traveling salesman problem, *International Journal of Computing and Intelligence Research*, 6 , pp. 475-484.

[36]. Ahmed ZH. (2015). A multi-parent genetic algorithm for the quadratic assignment problem, *Opsearch*, 52, pp. 714-7325.

[37]. Reinelt G. (1991). TSPLIB—A traveling salesman problem library, *ORSA Journal on Computing*, 3(4), pp. 376-384. (http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/ Accessed on 20th December 2023)

[38]. Ahmed ZH, Maleki F, Yousefikhoshbakht M, and Haron H. (2023). Solving the vehicle routing problem with time windows using modified football game algorithm, *Egyptian Informatics Journal*, 24(4), 100403.