Ahmed *Mathematical Sciences* 2013, **7**:10
http://www.iaumath.com/content/7/1/10

Mathematical Sciences
a SpringerOpen Journal

ORIGINAL RESEARCH Open Access

# An experimental study of a hybrid genetic algorithm for the maximum traveling salesman problem

Zakir Hussain Ahmed

## Abstract

**Purpose:** In this paper, we consider the maximum traveling salesman problem, a variation of the usual traveling salesman problem, in which the objective is to maximize the cost of a tour of the salesman. The main purpose of this paper is to develop a hybrid genetic algorithm (GA) for obtaining a heuristically optimal solution to the problem.

**Methods:** First, a simple GA and then a hybrid GA have been proposed to solve the problem. As crossover operator plays a vital role in GAs, we modify the sequential constructive crossover operator for our simple GA to solve the problem. To improve the quality of the solution obtained by the crossover operator, restricted 2-opt search is applied. Then a hybrid GA is developed by incorporating a new local search algorithm to the simple GA in order to obtain a heuristic solution to the problem.

**Results:** We compare the efficiency of our hybrid GA against an existing heuristic algorithm for symmetric traveling salesman problem library (TSPLIB) instances. Finally, we present solutions to the problem for asymmetric TSPLIB instances. Since, to the best of our knowledge, no literature presents solution for asymmetric instances, hence, we could not carry out any comparative study to show the efficiency of our hybrid GA for the asymmetric instances.

**Conclusions:** The comparative study shows the effectiveness of our hybrid GA.

**Keywords:** Maximum traveling salesman problem, Hybrid genetic algorithm, Sequential constructive crossover 2-Opt search, Local search

## Introduction

The traveling salesman problem (TSP) is a well-known problem in computer science and operations research. It has been studied for many years, and accordingly, many good algorithms have been developed to solve the problem. The maximum traveling salesman problem (Max-TSP) is a variation of the TSP in which the objective is to maximize the cost of a tour of the salesman. The problem can be defined as follows:

A network with $n$ nodes, being 'node 1' as the starting node, and a cost (or distance, or time, etc.) matrix C = $[c_{ij}]$ of order n associated with ordered pair of nodes (i, j) is given. The problem is to find a maximum cost Hamiltonian cycle. That is, the problem is to obtain a tour $(1 = \alpha_0, \alpha_1, \alpha_2, \ldots, \alpha_{n-1}, \alpha_n = 1) \approx \{1 \rightarrow \alpha_1 \rightarrow \alpha_2 \rightarrow \ldots \rightarrow \alpha_{n-1} \rightarrow 1\}$ for which the total cost $C(1 = \alpha_0, \alpha_1, \alpha_2, \ldots, \alpha_{n-1}, \alpha_n = 1) \approx \sum_{i=0}^{n-1} c(\alpha_i, \alpha_{i+1})$ is maximum.

It is well known that both the TSP and the Max-TSP are nondeterministic polynomial time (NP)-hard problems [1]. Of course, the Max-TSP can be reduced to the TSP (and vice versa); however, the special structure that leads to a well-solvable case for the TSP does not necessarily yield a well-solvable case for the Max-TSP. On the basis of the cost matrix structure, the Max-TSP is classified as symmetric (Max-STSP) or asymmetric (Max-ATSP). The problem is symmetric if $c_{ij} = c_{ji}$, $\forall$ $i$, $j$, and asymmetric otherwise. Also, the problem that satisfies the triangular inequality is called the metric problem. The Max-STSP is a special case of the Max-ATSP, and hence, the latter is found to be harder than the former. The Max-STSP is shown to be NP-hard [2], and

Correspondence: zhahmed@gmail.com
Department of Computer Science, Al-Imam Muhammad Ibn Saud Islamic University, P.O. Box No. 5701, Riyadh 11432, Kingdom of Saudi Arabia

the Max-ATSP is APX-hard [3]. Therefore, a polynomial time approximation is not desirable for the Max-TSP unless P = NP. A detailed study on the Max-TSP is carried out by Barvinok et al. [4]. The Max-TSP finds application in maximum latency delivery problems [5] and in the computation of the shortest common superstrings [6].

In this paper, we consider the general Max-TSP of both symmetric and asymmetric cases. Due to the complexity of Max-TSP, it is necessary to apply heuristics to solve instances of different sizes. Genetic algorithms (GAs) are one of the best heuristics that have been successfully applied to the TSP and its variations. This paper develops a hybrid GA using sequential constructive crossover [7], restricted 2-opt search, and a new local search algorithm to obtain a heuristic solution to the problem. We compare the efficiency of our hybrid algorithm against the heuristic algorithm of Fekete et al. [8] for symmetric traveling salesman problem library (TSPLIB) instances. The comparative study shows the effectiveness of our hybrid algorithm. Finally, we present solutions to the problem for asymmetric TSPLIB instances. Since, to the best of our knowledge, our results on Max-ATSP are the first in the literature, we could not carry out any comparative study for this case.

This paper is organized as follows: the next section provides a literature review, a hybrid genetic algorithm for the Max-TSP is presented in the 'Methods' section, presentation of computational experience for the hybrid algorithm is in the 'Results and discussion' section, and finally, comments and concluding remarks are presented in the 'Conclusions' section.

## Literature review

A number of different methods have been proposed for obtaining approximate solutions to the different cases of Max-TSP. For the Max-ATSP, Fisher et al. [9] developed a 1/2-approximation algorithm with polynomial time. The algorithm is then improved by Kosaraju et al. [10] who developed a polynomial approximation algorithm with a performance ratio of 38/63. Lewenstein and Sviridenko [11] developed a better approximation algorithm with a performance ratio of 5/8. An $O(n^3)$-time polynomial approximation algorithm that achieves an approximation ratio of 8/13 has been developed [3]. Kaplan et al. [12] proposed an approximation algorithm that achieves an approximation guarantee of 2/3. An approximation algorithm with a performance ratio of 31/40 has been developed for the metric Max-ATSP [13]. Currently, Kowalik and Mucha [14] developed an approximation algorithm with the best approximation ratio 35/44 for metric Max-ATSP. Paluch et al. [15] proposed a simple approximation algorithm for the Max-ATSP which guarantees an approximation of 2/3; however, it matches the approximation guaranteed by Kaplan et al. [12].

For the Max-STSP, approximation algorithms with various performance ratios have been developed [16,17]. Kowalik and Mucha [18] developed an approximation algorithm with a performance ratio of 7/8 for metric Max-TSP.

Several researchers investigated the Max-TSP on special matrices. Deineko and Woeginger [19] investigated the Max-TSP on symmetric Demidenko matrices and found that in strong contrast to the usual TSP, the Max-TSP is NP-hard to solve. They identify several special cases that are solvable in polynomial time. Blokh and Levner [20] investigated the properties of the Max-TSP on nonnegative quasi-banded matrices, and they proved that it is strongly NP-hard and derived a linear-time approximation algorithm with a guaranteed performance. Steiner and Xue [21] investigated the Max-TSP on van der Veen matrices and established that the problem stays NP-hard even on the class of distance matrices which satisfy *both* the van der Veen and Demidenko conditions.

However, all of the above studies do not provide any computational experience for the problems. Also, most of the literatures discussed above deal with only a particular case of Max-TSP, whereas our proposed algorithms are capable of dealing with all cases of Max-TSP without any modification of the algorithms. Fekete et al. [8] developed a heuristic algorithm for solving the Max-STSP and reported computational experience for symmetric TSPLIB instances only. We are going to compare our results with the results of Fekete et al. [8] for symmetric TSPLIB instances only.

## Methods

GAs have been used widely to deal with the usual TSP. They are based on mimicking the survival of the fittest among species generated by random changes in the gene structure of the chromosomes in evolutionary biology [22]. They start with a set of chromosomes called initial population and then go through (possibly) three operations, namely reproduction/selection, crossover, and mutation, to obtain a heuristically optimal solution.

### Initial population

There are various ways to represent a solution by a chromosome in GAs for the TSP and its variations. We consider the path/order representation for a chromosome that simply lists the nodes for a Max-TSP instance. We also consider a randomly generated feasible set of chromosomes of fixed size as the initial population for our GA.

### Fitness function and reproduction

The fitness function is the cost of a tour represented by a chromosome. In reproduction, no new chromosome is created; some chromosomes are copied to the next generation probabilistically based on their fitness values. In our GA, the stochastic remainder selection method [23] is used for reproduction.

### Sequential constructive crossover operator

The crossover operation selects a pair of parent chromosomes and exchanges their information. Since crossover is the most important operator in GAs, various crossover operators have been proposed for the usual TSP [22]. The sequential constructive crossover (SCX) [7] is found to be one of the best crossover operators. It produces only one offspring from a pair of parents. It has been applied to the bottleneck traveling salesman problem and found good results [24]. Here also, we consider this SCX for our GA. However, we slightly modify the operator to fit to our problem as follows:

- Step 1: Start from ‘node 1’ (i.e., current node p = 1).
- Step 2: Sequentially search both of the parent chromosomes and consider the first ‘legitimate node’ (the node that does not appear in the present incomplete offspring chromosome) that appeared after ‘node p’ in each parent. If no legitimate node after ‘node p’ is present in any of the parents, search sequentially from the starting of the parent and consider the first legitimate node and go to step 3.
- Step 3: Suppose the ‘node α’ and the ‘node β’ are found in the 1st and the 2nd parent, respectively, then for selecting the next node, go to step 4.
- Step 4: If $c_{p\alpha} > c_{p\beta}$, then select node α, otherwise node β, as the next node and concatenate it to the partially constructed offspring chromosome. If the offspring is a complete chromosome, then stop; otherwise, rename the present node as node p and go to step 2.

Let a pair of selected chromosomes be $P_1$: (1, 5, 7, 3, 6, 4, 2) and $P_2$: (1, 4, 5, 2, 6, 3, 7) with values 312 and 335, respectively, with respect to the cost matrix given in Table 1. By applying the above SCX, we obtain the offspring (1, 5, 7, 4, 2, 3, 6) with the value 376, which is larger and better than both parents.

**Table 1 The cost matrix**

| Node | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| 1 | −999 | 75 | 99 | 9 | 35 | 63 | 8 |
| 2 | 51 | −999 | 86 | 46 | 88 | 29 | 20 |
| 3 | 100 | 5 | −999 | 16 | 28 | 35 | 28 |
| 4 | 20 | 45 | 11 | −999 | 59 | 53 | 49 |
| 5 | 86 | 63 | 33 | 65 | −999 | 76 | 72 |
| 6 | 36 | 53 | 89 | 31 | 21 | −999 | 52 |
| 7 | 58 | 31 | 43 | 67 | 52 | 60 | −999 |

For this crossover operation, a pair of parents is selected sequentially from the mating pool, and only one offspring is produced. In order to avoid performing the crossover operation of same parent chromosomes, we check whether the chromosomes are the same. If they are found to be the same, some of the genes (nodes) of the second parent chromosome are exchanged temporarily only for the crossover, and then we go for the crossover operation. The present second original parent will then be the first parent for the next crossover operation when pairing with the next chromosome in order and so on. To improve the quality of the solution by SCX, we follow the following method. If the offspring is better than the parent, the 2-opt search is applied to the offspring to improve further, and then the first parent is replaced by the improved offspring.

### Mutation operation

The mutation operation is the occasional random alteration of the genes in a chromosome. By performing occasional random changes in the chromosomes, GAs ensure that new parts of the search space are reached, which reproduction and crossover cannot fully guarantee. In doing so, mutation ensures that no important features are prematurely lost, thus maintaining the mating pool diversity. For this investigation, we have considered the reciprocal exchange mutation, which selects two genes randomly and swaps them.

Our simple GA works by randomly generating an initial population of strings, which is referred to as gene pool, and then applying the above reproduction, crossover, and mutation operators to create new, and hopefully, better populations as successive ‘generations.’ Simple GAs focus on the global aspects of an optimization task, whereas local search methods focus on the local aspects of the optimization task. The hybridization of both genetic algorithm and local search methods has shown to be an effective route to follow for finding high-quality solutions for combinatorial optimization problems [25]. Most of the hybrid GAs in the literatures are developed by incorporating 2-opt, Or-opt, 3-opt, and LK local search heuristics to the simple GAs [25]. For our hybrid GA for the Max-TSP, the following local search algorithm is developed and incorporated to the simple GA.

### Local search algorithm

The proposed local search algorithm is basically a combined mutation operator that combines three mutation operators - insertion, inversion, and reciprocal exchange, with cent per cent of probabilities, which has been proposed for the bottleneck TSP and applied to the best tour obtained by a sequential constructive sampling algorithm [26]. We modify the operator for our problem and apply to the present best tour found so far. The algorithm is as follows: Suppose the present best tour is $(1 = \beta_0, \beta_1, \beta_2, \ldots, \beta_{n-1})$, then the local search algorithm can be developed as follows:

- Step 1: For $i = 1$ to $n - 2$, do the following steps.
- Step 2: For $j = i + 1$ to $n - 1$, do the following steps.
- Step 3: If inserting node $\beta_i$ after node $\beta_j$ improves the present tour value, then insert node $\beta_i$ after node $\beta_j$. In any case, go to step 4.
- Step 4: If inverting substring between nodes $\beta_i$ and $\beta_j$ improves the present tour value, then invert the substring. In any case, go to step 5.
- Step 5: If swapping nodes $\beta_i$ and $\beta_j$ improves the present tour value, then swap them.

Our hybrid genetic algorithm (HGA) may be summarized as in Figure 1.

## Results and discussion

Our HGA has been encoded in Visual C++ on a Pentium IV personal computer, with a speed of 3 GHz and a 448 MB RAM under MS Windows XP operating system and is tested with TSPLIB [27] instances. The following parameters are selected for our algorithm: population size is 100, crossover probability is 1.0, maximum of 20,000 generations as termination condition, and 20 independent runs for each setting. For setting mutation probability, five mutation probabilities: 0, 0.01, 0.02, 0.03, and 0.04, have been applied on five instances.

Table 2 reports the mean and standard deviation (SD) of the best solution values over 20 trials for five mutation probabilities on instances eil101, bier127, ch150, gil262, and a280. The HGA using mutation probabilities from 0.01 to 0.03 can lead to significant improvements over HGA without a mutation operator on these five test instances. The table clearly indicates that the mutation also plays an important role in obtaining good solution to these instances. The significance of this improvement is further proved by the statistical one-tailed $t$ test with a confidence level of 0.05. Table 2 also shows the $p$ values of the $t$ test with a confidence level of 0.05 on the best solution value between HGA without a mutation operator and with a mutation operator using different mutation probabilities for the instances. The italics denotes the significant improvements. It is seen that as the mutation probability increases, the quality of the solutions decreases, and for two instances, HGA with a probability of 0.04 obtains worse solutions than the HGA without a mutation operator. However, the HGA with a mutation probability of 0.01 is found to obtain the best solutions, which would be considered for later study.

Table 3 summarizes the results for some moderate-sized symmetric TSPLIB instances of sizes up to 417 only. For these instances, we compare the efficiency of our HGA against the heuristic algorithm (CROSS + Lin-Kernighan, therein) of Fekete et al. [8] in terms of solution quality. We also report the optimal solutions by Fekete et al. [8] using CONCORDE code [28]. We report the best and worst solution values in 20 runs. Also, we report the percentage of error of the average solution values as Error (%) = (Opt − Average) / Opt × 100%, where Opt is the optimal solution value calculated using CONCORDE reported in [8]. The table also reports the percentage of error of the best solutions obtained by Fekete et al. [8].

From Table 3, it is very clear that our HGA is better than algorithm by Fekete et al. [8] for the symmetric instances, in terms of solution quality. Also, on average, the solutions are at most 0.017% away from the optimal solutions, which is very good. Treating this study as a base for the effectiveness of our HGA, we can now present solutions for the asymmetric TSPLIB instances. It is to be noted that our HGA does not require any modification for solving different types and cases of TSPLIB instances, whereas algorithm by Fekete et al. [8] and CONCORDE code [28] require modifications.

Table 4 summarizes the results for 28 asymmetric TSPLIB instances of sizes up to 443. Also, the average of complete computational times and the times when the final solutions are seen for the first time are reported (in seconds) in 20 runs. For half of the instances, the solution quality is found to be insensitive to the number of runs. These instances are of sizes less than 130, and most of them are ‘ftv’ instances. For instances ry48p of size 48, ft53 of size 53, ft70 of size 70, and kro124p of size 100, the solution quality is found to be sensitive to the number of runs. We can say that these instances as well as the instances of sizes more than 130 are hard. Since no literature presents optimal solutions for the asymmetric instances, so to measure the quality of the solutions, we report the average percentage of errors from the best solution values among 20 runs. For these instances, the average percentage of error ranges from 0.0000% to 0.1257%, which is not bad. On the basis of computational time, on average, the algorithm finds a final solution for the first time within only 13% of complete computational times. That is, on average, for these instances, the algorithm finds optimal solutions in the beginning of the generations. Moreover, for the four ‘rbg’ instances, the algorithm finds the optimal solutions in the very beginning of the iteration, within 1% of complete computational times.

## Conclusions

We presented a simple GA using the sequential constructive crossover operator to obtain a heuristic

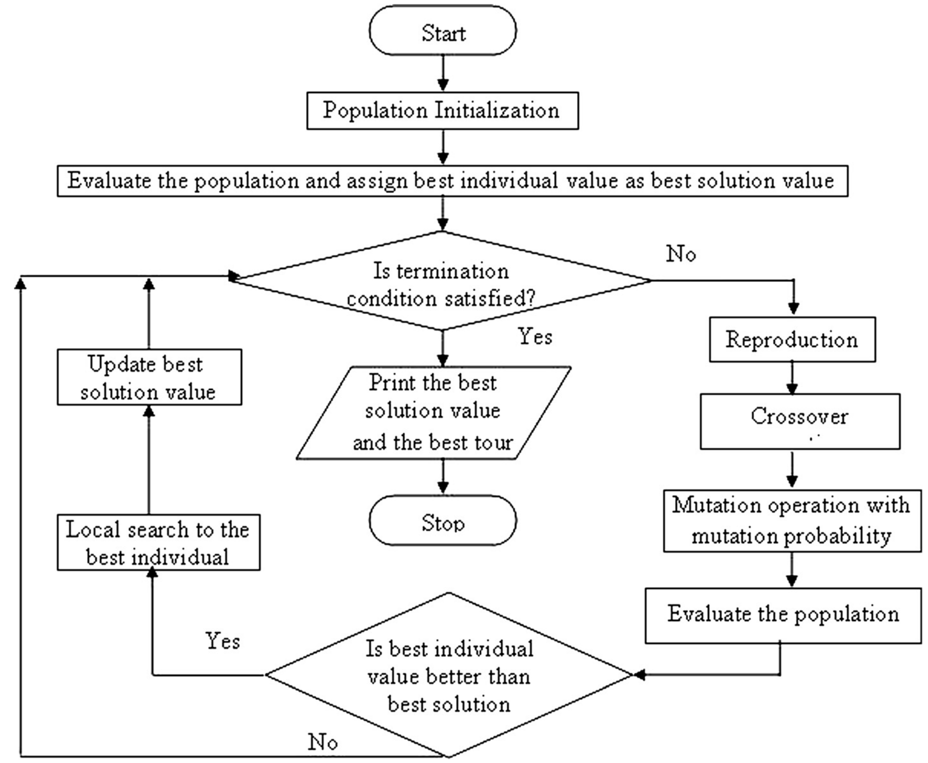
Start
Population Initialization
Evaluate the population and assign best individual value as best solution value
Is termination condition satisfied?
No
Yes
Reproduction
Update best solution value
Print the best solution value and the best tour
Crossover
Stop
Mutation operation with mutation probability
Local search to the best individual
Evaluate the population
Yes
Is best individual value better than best solution
No

**Figure 1** **Flow chart of our hybrid genetic algorithm.**

solution to the Max-TSP. The restricted 2-opt search and a new local search algorithm that combines three mutation operators with cent per cent of mutation probability have been incorporated for hybridizing the simple GA for obtaining better solutions for the problem. We then compared the efficiency of our hybrid algorithm against the heuristic algorithm of Fekete et al. [8] for the symmetric TSPLIB instances and then presented solutions to the problem for 28 asymmetric TSPLIB instances of sizes up to 443. The computational

**Table 2 Results over five different instances using different mutation probabilities**

| Instance | | $P_m = 0$ | $P_m = 0.01$ | $P_m = 0.02$ | $P_m = 0.03$ | $P_m = 0.04$ |
|---|---|---|---|---|---|---|
| eil101 | Mean | 4,978.20 | 4,980.00 | 4,980.00 | 4,980.00 | 4,980.00 |
| | SD | 0.92 | 0.00 | 0.00 | 0.00 | 0.00 |
| | *P* value | - | *8.00E−05* | *8.00E−05* | *8.00E−05* | *8.00E−05* |
| bier127 | Mean | 840,804.20 | 840,815.00 | 840,814.70 | 840,813.30 | 840,803.20 |
| | SD | 9.82 | 0.00 | 0.67 | 2.75 | 6.14 |
| | *P* value | - | *3.48E−03* | *3.60E−03* | *7.06E−03* | 3.82E−01 |
| ch150 | Mean | 78,565.60 | 78,571.00 | 78,569.80 | 78,566.60 | 78,566.00 |
| | SD | (0.84) | (0.00) | (0.42) | (0.52) | (1.33) |
| | *P* value | - | *4.07E−09* | *2.06E−08* | *1.14E−02* | *1.72E−01* |
| gil262 | Mean | 39,211.20 | 39,223.00 | 39,218.40 | 39,215.20 | 39,210.00 |
| | SD | 1.03 | 0.82 | 0.84 | 1.03 | 1.63 |
| | *P* value | - | *2.06E−10* | *2.76E−08* | *4.39E−05* | 7.02E−02 |
| a280 | Mean | 50,680.40 | 50,694.70 | 50,691.50 | 50,689.60 | 50,680.80 |
| | SD | 0.84 | 0.48 | 1.35 | 3.10 | 2.78 |
| | *P* value | - | *9.28E−14* | *3.98E−09* | *5.85E−06* | *3.26E−01* |

**Table 3 Results for symmetric TSPLIB instances**

| Instance | *n* | Optimal solution | Fekete et al. [8] | | Solution by HGA | | | |
|---|---|---|---|---|---|---|---|---|
| | | | Solution | Error (%) | Best | Worst | Average | Error (%) |
| eil101 | 101 | 4,980 | 4,966 | 0.2811 | 4,980 | 4,980 | 4,980.00 | 0.0000 |
| bier127 | 127 | 840,815 | 840,810 | 0.0006 | 840,815 | 840,815 | 840,815.00 | 0.0000 |
| ch150 | 150 | 78,571 | 78,552 | 0.0242 | 78,571 | 78,571 | 78,571.00 | 0.0000 |
| gil262 | 262 | 39,229 | 39,170 | 0.1504 | 39,224 | 39,222 | 39,223.00 | 0.0170 |
| a280 | 280 | 50,702 | 50,638 | 0.1262 | 50,695 | 50,694 | 50,694.70 | 0.0130 |
| lin318 | 318 | 860,512 | 860,464 | 0.0056 | 860,503 | 860,496 | 860,499.90 | 0.0000 |
| rd400 | 400 | 311,732 | 311,648 | 0.0269 | 311,720 | 311,719 | 311,719.50 | 0.0000 |
| fl417 | 417 | 779,331 | 779,236 | 0.0122 | 779,316 | 779,315 | 779,315.25 | 0.0000 |

experience shows that our algorithm is better than the algorithm of Fekete et al. [8] for symmetric instances. Since, to the best of our knowledge, no literature presents solutions for asymmetric instances, hence, we could not carry out any comparative study to show the efficiency of the algorithm for the asymmetric instances. Since for the symmetric instances our algorithm finds very good solutions, we hope that the reported solutions for the asymmetric instances are very close to the exact optimal solutions, if not exact.

**Table 4 Results for asymmetric TSPLIB instances**

| Instance | n | Solution | | | | Average time | |
|---|---|---|---|---|---|---|---|
| | | Best | Worst | Average | Error(%) | First seen | Complete |
| br17 | 17 | 445 | 445 | 445.00 | 0.0000 | 0.00 | 8.83 |
| ftv33 | 34 | 6,006 | 6,006 | 6,006.00 | 0.0000 | 5.81 | 19.33 |
| ftv35 | 36 | 6,693 | 6,693 | 6,693.00 | 0.0000 | 6.82 | 25.17 |
| ftv38 | 39 | 7,136 | 7,136 | 7,136.00 | 0.0000 | 6.03 | 30.57 |
| p43 | 43 | 29,077 | 29,077 | 29,077.00 | 0.0000 | 7.31 | 29.89 |
| ftv44 | 45 | 8,668 | 8,668 | 8,668.00 | 0.0000 | 15.25 | 37.43 |
| ftv47 | 48 | 9,502 | 9,502 | 9,502.00 | 0.0000 | 22.12 | 45.12 |
| ry48p | 48 | 78,122 | 78,001 | 78,047.45 | 0.0954 | 35.10 | 47.64 |
| ft53 | 53 | 34,966 | 34,921 | 34,936.10 | 0.0855 | 42.86 | 54.25 |
| ftv55 | 56 | 10,273 | 10,273 | 10,273.00 | 0.0000 | 22.33 | 53.13 |
| ftv64 | 65 | 12,216 | 12,216 | 12,216.00 | 0.0000 | 28.69 | 66.90 |
| ft70 | 70 | 91,562 | 91,366 | 91,499.26 | 0.0685 | 53.83 | 81.65 |
| ftv70 | 71 | 13,613 | 13,613 | 13,613.00 | 0.0000 | 39.97 | 77.70 |
| ftv80 | 81 | 12,721 | 12,721 | 12,721.00 | 0.0000 | 4.57 | 95.53 |
| ftv90 | 91 | 15,023 | 15,023 | 15,023.00 | 0.0000 | 1.26 | 118.22 |
| kro124p | 100 | 286,311 | 285,999 | 286,075.90 | 0.0821 | 105.22 | 141.10 |
| ftv100 | 101 | 18,266 | 18,266 | 18,266.00 | 0.0000 | 2.05 | 148.50 |
| ftv110 | 111 | 21,277 | 21,277 | 21,277.00 | 0.0000 | 2.45 | 177.06 |
| ftv120 | 121 | 24,277 | 24,277 | 24,277.00 | 0.0000 | 15.36 | 188.16 |
| ftv130 | 131 | 27,745 | 27,722 | 27,731.10 | 0.0501 | 75.66 | 210.15 |
| ftv140 | 141 | 30,481 | 30,443 | 30,460.70 | 0.0666 | 118.13 | 251.33 |
| ftv150 | 151 | 33,753 | 33,720 | 33,738.70 | 0.0424 | 193.91 | 298.97 |
| ftv160 | 161 | 36,098 | 36,085 | 36,092.20 | 0.0161 | 92.97 | 319.71 |
| ftv170 | 171 | 38,439 | 38,420 | 38,427.82 | 0.0291 | 106.71 | 342.06 |
| rbg323 | 323 | 8,253 | 8,251 | 8,251.90 | 0.0133 | 12.15 | 1,165.71 |
| rbg358 | 358 | 9,315 | 9,301 | 9,313.20 | 0.0193 | 15.20 | 1,316.08 |
| rbg403 | 403 | 10,223 | 10,207 | 10,210.15 | 0.1257 | 16.05 | 1,480.70 |
| rbg443 | 443 | 10,955 | 10,932 | 10,944.16 | 0.099 | 21.13 | 1,715.93 |

### Competing interests

The author declares that he has no competing interests.

### Acknowledgements

The author is thankful to the honorable anonymous reviewers for their constructive comments and suggestions. This research was partially supported by King Abdulaziz City for Science and Technology, Saudi Arabia, vide grant no. 11-INF1788-08.

Received: 2 June 2012 Accepted: 24 January 2013
Published: 12 February 2013

### References

1. Lawler, EL, Lenstra, JK, Rinnooy Kan, AHG, Shmoys, DB: The Travelling Salesman Problem. Wiley, Chichester (1985)
2. Garey, MR, Johnson, DS: Computer and Intractability: A Guide to the Theory of NP-Completeness. Freeman, New York (1979)
3. Bläser, M: An 8/13-approximation algorithm for the asymmetric maximum TSP. Journal of Algorithms **50**(1), 23–48 (2004)
4. Barvinok, A, Gimaldi, E, Serdyukov, A: The maximum TSP. In: Gutin, G, Punnen, A (eds.) The Traveling Salesman Problem and Its Variations, pp. 585–607. Kluwer, Dordrecht (2002)
5. Chalasani, P, Motwani, R: Approximating capacitated routing and delivery problems. SIAM Journal of Computing **28**, 2133–2149 (1999)
6. Breslauer, D, Jiang, T, Jiang, Z: Rotations of periodic strings and short superstrings. Journal of Algorithms **24**, 340–353 (1997)
7. Ahmed, ZH: Genetic algorithm for the traveling salesman problem using sequential constructive crossover. International Journal of Biometrics & Bioinformatics **3**(6), 96–105 (2010)
8. Fekete, SP, Meijer, H, Rohe, A, Tietze, W: Solving a "hard" problem to approximate an "easy" one: heuristics for maximum matchings and maximum traveling salesman problems. In: Buchsbaum, AL, Snoeyink, J (eds.) ALENEX, LNCS 2153, pp. 1–16. Springer, Berlin (2001)
9. Fisher, ML, Nemhauser, L, Wolsey, LA: An analysis of approximations for finding a maximum weight Hamiltonian circuit. Networks **12**(1), 799–809 (1979)
10. Kosaraju, SR, Park, JK, Stein, C: Long tours and short superstrings. In: Proceedings of the 35th Annual IEEE Symposium on Foundations of Computer Science, pp. 166–177., Santa Fe (1994). 20–22 Nov
11. Lewenstein, M, Sviridenko, M: A 5/8-approximation algorithm for the maximum asymmetric TSP. SIAM Journal of Discrete Mathematics **17**(2), 237–248 (2003)
12. Kaplan, H, Lewenstein, M, Shafrir, N, Sviridenko, M: Approximation algorithms for asymmetric tsp by decomposing directed regular multigraphs. Journal of ACM **52**(4), 602–626 (2005)
13. Bläser, M, Ram, LS, Sviridenko, M: Improved approximation algorithms for metric maximum ATSP and maximum 3-cycle cover problems. Operations Research Letters **37**, 176–180 (2009)
14. Kowalik, L, Mucha, M: 35/44-approximation for asymmetric maximum tsp with triangle inequality. Algorithmica **59**(2), 240–255 (2011)
15. Paluch, K, Elbassioni, K, van Zuylen, A: Simpler approximation of the maximum asymmetric traveling salesman problem. In: Christoph, D, Thomas, W (eds.) 29th Symposium on Theoretical Aspects of Computer Science (STACS'12), pp. 501–506. Dagstuhl, Wadern (2012)
16. Hassin, R, Rubinstein, S: Better approximations for max TSP. Information Processing Letters **75**(4), 251–258 (2000)
17. Chen, Z-Z, Okamoto, Y, Wang, L: Improved deterministic approximation algorithms for Max TSP. Information Processing Letters **95**, 333–342 (2005)
18. Kowalik, L, Mucha, M: Deterministic 7/8-approximation for the metric maximum TSP. Theoretical Computer Science **410**(47–49), 5000–5009 (2009)
19. Deineko, VG, Woeginger, GJ: The Maximum Travelling Salesman Problem on symmetric Demidenko matrices. Discrete Applied Mathematics **99**, 413–425 (2000)
20. Blokh, D, Levner, E: An approximation algorithm with performance guarantees for the maximum traveling salesman problem on special matrices. Discrete Applied Mathematics **119**, 139–148 (2002)
21. Steiner, G, Xue, Z: The maximum traveling salesman problem on van der Veen matrices. Discrete Applied Mathematics **146**, 1–2 (2005)
22. Goldberg, DE: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, New York (1989)
23. Deb, K: Optimization for Engineering Design: Algorithms and Examples. PHI, New Delhi (1995)
24. Ahmed, ZH: A hybrid genetic algorithm for the bottleneck traveling salesman problem. ACM Transactions on Embedded Computing Systems **12**(1), 2013 (2013). In: in press
25. Michalewicz, Z: Genetic Algorithms + Data Structures = Evolution Problems, 2nd edn. Springer, New York (1994)
26. Ahmed, ZH: A hybrid sequential constructive sampling algorithm for the bottleneck traveling salesman problem. International Journal of Computational Intelligence Research **6**(3), 475–484 (2010)
27. Reinelt, G, TSPLIB: http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/ (2013). Accessed 30 Jan 2013
28. Cook, W, CONCORDE: http://www.tsp.gatech.edu/concorde/index.html (2013) Accessed 30 Jan 2013

doi:10.1186/2251-7456-7-10
**Cite this article as:** Ahmed: **An experimental study of a hybrid genetic algorithm for the maximum traveling salesman problem.** *Mathematical Sciences* 2013 **7**:10.