MGA-TSP: Modernized Genetic Algorithm for the Traveling Salesman Problem

Article *in* International Journal of Reasoning-based Intelligent Systems · January 2019

DOI: 10.1504/IJRIS.2019.10019776

7 authors, including:

Ra'ed M. Al-Khatib
Yarmouk University
25 PUBLICATIONS **117** CITATIONS
SEE PROFILE

Mohammed Azmi Al-Betar
Ajman University
221 PUBLICATIONS **5,111** CITATIONS
SEE PROFILE

Mohammed a. Awadallah
Al-Aqsa University
98 PUBLICATIONS **2,007** CITATIONS
SEE PROFILE

Khalid Nahar
Yarmouk University
60 PUBLICATIONS **307** CITATIONS
SEE PROFILE

MGA-TSP: modernised genetic algorithm for the travelling salesman problem

Ra'ed M. Al-Khatib*

Department of Computer Sciences,
Faculty of Information Technology and Computer Sciences,
Yarmouk University,
Irbid-21163, Jordan
Email: raed.m.alkhatib@yu.edu.jo
*Corresponding author

Mohammed Azmi Al-Betar

Department of Information Technology,
Al-Huson University College,
Al-Balqa Applied University (BAU),
P.O. Box 50, Al-Huson,
Irbid, Jordan
Email: mohbetar@bau.edu.jo

Mohammed A. Awadallah

Department of Computer Science,
Al-Aqsa University,
P.O. Box 4051, Gaza, Palestine
Email: ma.awadallah@alaqsa.edu.ps

Khalid M.O. Nahar

Department of Computer Sciences,
Faculty of Information Technology and Computer Sciences,
Yarmouk University,
Irbid-21163, Jordan
Email: khalids@yu.edu.jo

Mohammed M. Abu Shquier

Faculty of Computer Science and Information Technology,
Jerash University,
Jerash, Jordan
Email: shquier@jpu.edu.jo

Ahmad M. Manasrah

Network and Information Security Department,
Faculty of Information Technology and Computer Sciences,
Yarmouk University,
Irbid-21163, Jordan
Email: ahmad.a@yu.edu.jo

Ahmad Bany Doumi

Department of Computer Sciences,
Faculty of Information Technology and Computer Sciences,
Yarmouk University,
Irbid-21163, Jordan
Email: ahmad.domi.usm@gmail.com

Abstract: This paper proposes a new enhanced algorithm called modernised genetic algorithm for solving the travelling salesman problem (MGA-TSP). Recently, the most successful evolutionary algorithm used for TSP problem, is GA algorithm. The main obstacles for GA

is building its initial population. Therefore, in this paper, three neighbourhood structures (*inverse*, *insert*, and *swap*) along with 2-*opt* is utilised to build strong initial population. Additionally, the main operators (i.e., crossover and mutation) of GA during the generation process are also enhanced for TSP. Therefore, powerful crossover operator called EAX is utilised in the proposed MGA-TSP to enhance its convergence. For validation purpose, we used TSP datasets, range from 150 to 33,810 cities. Initially, the impact of each neighbouring structure on the performance of MGA-TSP is studied. In conclusion, MGA-TSP achieved the best results. For comparative evaluation. MGA-TSP is able to outperform six comparative methods in almost all TSP instances used.

Keywords: travelling salesman problem; optimisation; genetic algorithm; neighbouring operators.

Reference to this paper should be made as follows: Al-Khatib, R.M., Al-Betar, M.A., Awadallah, M.A., Nahar, K.M.O., Abu Shquier, M.M., Manasrah, A.M. and Doumi, A.B. (2019) 'MGA-TSP: modernised genetic algorithm for the travelling salesman problem', *Int. J. Reasoning-based Intelligent Systems*, Vol. 11, No. 3, pp.215–226.

Biographical notes: Ra'ed M. Al-Khatib obtained his PhD in Computer Science from the Universiti Sains Malaysia, Malaysia, in 2012. He joined the Faculty of Information Technology and Computer Sciences (FITACS), Yarmouk University (YU) from February 2016. His main research interests include artificial intelligence (AI), natural language processing (NLP) and machine learning. He is interested in conducting research in the areas of computational bioinformatics, information retrieval, IoTs, high parallel computing (HPC), biometrics and Arabic NLP in general. In addition to that he conducted good research in the area of wireless sensor networks (WSNs).

Mohammed Azmi Al-Betar is an Associated Professor in the Department of Computer Science at Al-Huson University College, Al-Balqa Applied University, and a senior member of Computational Intelligence Research Group in the School of Computer Science at the University Sains Malaysia (USM). He received his BSc and MSc from the Computer Science Department at the Yarmouk University, Irbid, Jordan in 2001 and 2003, respectively. He obtained his PhD from the School of Computer Sciences at the University Sains Malaysia (USM), Pulau Penang, Malaysia in October 2010. He has published a number of high quality papers in international journals and conferences. His research interests are mainly directed to metaheuristic optimisation methods and hard combinatorial optimisation problems including scheduling and timetabling.

Mohammed A. Awadallah received his PhD from the School of Computer Sciences at the Universiti Sains Malaysia (USM), Pulau Penang, Malaysia, in 2014. He is currently an Assistant Professor in the Department of Computer Science, Al-Aqsa University, P.O. Box 4051, Gaza, Palestine. Since July 2018, he is the Dean of the Faculty of Computers and Information Technology. He has authored many publications in international journals, conferences, and book chapters. His research interests on optimisation methods and combinatorial optimisation problems.

Khalid M.O. Nahar obtained his PhD in Computer Science and Engineering from the King Fahd University of Petroleum and Minerals (KFUPM), KSA, in 2013. He is currently working at the Yarmouk University-Jordan, as an Assistant Professor in the Department of Computer Sciences. He is currently the Dean Assistant for Quality Control. His research interests are continuous speech recognition, Arabic computing, natural language processing, wireless sensor networks (WSNs), multimedia computing, content-based retrieval, artificial intelligence, and software engineering.

Mohammed M. Abu Shquier obtained his PhD in Computer Science from the Universiti Sains Malaysia, Malaysia, in 2008. His research interests focuses on developing novel Arabic machine translation (rule and statistical-based). He also interested in conducting research in the areas of computational linguistics, information retrieval and arabic natural language processing in general. In addition to that he had conducted good research in the area of Arabic morphology, syntactic and symantec.

Ahmad M. Manasrah is currently an Associate Professor at the Yarmouk University, Network and Information Security Department, Jordan since 2011. prior to joining Yarmouk University, He was the Deputy Director of Research and Innovation at the National Advanced IPv6 Center, Malaysia for two years. He has published over 30 international journal and 11 research papers including proceedings and three book chapters. He received his PhD from the Universiti Sains Malaysia (USM) in 2009. His research interest includes advanced internet security and network monitoring.

Ahmad Bany Doumi is currently a PhD student at the Jordan University (JU), Amman-Jordan. He finished his Master in Computer Science from the Yarmouk University in 2015.

1 Introduction

Travelling salesman problem (TSP) is traditional combinatorial optimisation problem belongs to NP-hard class in almost all of its variations (Michael and David, 1979). TSP is tackled by finding an optimal tour by means of visiting a given cities each of which visited exactly once and the salesman returns to the initial city as minimum distance as possible (Dorigo and Gambardella, 1997; Nagata and Kobayashi, 2013). The complexity of TSP is normally increased when the number of cities to be visited is increased (Helsgaun, 2000). TSP are normally used as a benchmark problem to evaluate the performance of newly established methods. TSP is very useful for real-world applications employed in traffic and military domains (Yoon and Cho, 2011; Starkey et al., 2016). Therefore, the attention of the researchers in the optimisation domain tends to utilise the approximation mechanisms to tackle TSP problems (Albayrak and Allahverdi, 2011).

Many methods either *exact* or *approximation* solutions, have been introduced for TSP. Although, *exact* methods can be efficiently works for small scaled TSP, they are not workable for the large scaled TSP problem due to the cost of the computational time required (Helsgaun, 2000; Nagata and Kobayashi, 2013). Several exact methods introduced for TSP such that 'concorde' solver, which efficiently tackled TSP up to 1,000 cities (Helsgaun, 2000; Applegate et al., 2011; Hoos and Stützle, 2014). The attention of the TSP-solver have been paid to the *approximation* methods due to their strength in finding a quasi-optimal solution for large scaled TSP with a reasonable computational time (Matai et al., 2010). A comprehensive and exhaustive survey of the approximation methods used for TSP problem are shown in Johnson and McGeoch (1997).

The *approximation* methods for TSP can be categorised into *heuristic* and *metaheuristic* methods. *Heuristic* methods are considered as a problem-dependent approach, which construct a solution for TSP from scratch, where their solution quality is not always taking into account (Geem et al., 2001; Helsgaun, 2000). *Heuristic* methods are normally used for several optimisation problems as a general optimisation framework. They are utilising learning operators control by given parameters to efficiently explore the problem search space and exploit the accumulative search (Pourhassan and Neumann, 2015). The *metaheuristic* methods are conventionally categorised into local search-based algorithms, evolutionary-based algorithms and swarm-based algorithms (Chen and Chien, 2011; Krause et al., 2013; Kong et al., 2006). Local search-based algorithms begins with random solution normally constructed by a heuristic method. They iteratively modify that solution using neighbouring mechanisms until a local optimal solution, which is normally in the same search space region as initial one is reached. Local search-based methods have been adapted for TSP including: Tabu search (Gendreau et al., 1998; Basu et al., 2017), simulated annealing (SA) (Bayram and Şahin, 2013), variable neighbourhood search (Mladenović et al., 2013). The evolutionary-based algorithms (EAs), however, begins with a set of random solutions called population. Generation after generation, new solutions are generated using recombination and mutation operators. The new offspring population normally replaced the parent population, if they are better. This evolution process will continued until stagnation point is reached. The popular EAs solved TSP include genetic algorithm (GA) (Maity et al., 2016; Changdar et al., 2014) and harmony search algorithm (HSA) (Wang et al., 2016). The swarm-based algorithms imitate a part of social living of some animals. They are initiated with a set of solution called swarm. In each iteration, these swarms are interact in a particular learning mechanism using self-organisation and decentralised control to come up with a new swarms. The most popular swarm-based algorithms used for TSP are ant colony optimiser (Liu et al., 2017; Kalyani, 2015), particle swarm optimisation (PSO) (Anantathanavit and Munlin, 2016), artificial bee colony (ABC) (Li et al., 2012), ant colony optimisation (ACO) algorithm (Chen et al., 2012), Bat algorithm (BA) (Saji and Riffi, 2016), and intelligent water drop (IWD) algorithm (Ouaarab et al., 2014; Alijla et al., 2014).

The most popular ground algorithm for TSP is GA, which is the focal point of the present paper (Zhao et al., 2008; Yi and Fang, 2010; Matei and Pop, 2010; Vahdati et al., 2010; Nagata and Kobayashi, 2013; Tsai et al., 2014; Senthilkumar and Prasanna, 2014; Thanh et al., 2015; Maity et al., 2016; Lin et al., 2016; Alipour et al., 2017). The studied of GA in the chronic shortcomings of GA are twofold:

1 The tendency of exploring several search space region at the same time but without deep search in each region.

2 Since it deals with a population of individuals, the required computational time is high.

Therefore, the TSP researchers turned their attention to employ local search-based algorithms due to their capabilities in local exploitation. However, up to now, the best recorded results achieved for TSP problem is obtained by GA (Nagata and Kobayashi, 2013). In this paper, GA is tailored to TSP with new neighbouring operators (*inverse*, *insert*, and *swap*), which are used to modernised the GA algorithm. This enhancement to GA is done in order to overcome the above two fold shortcomings of initial GA algorithm. Basically, the main stages of the GA-based algorithm for TSP, are *selection*, *crossover*, and *mutation* operators.

There are many GA-based algorithms have been developed for TSP. The research focus on modifying a new *selection*, *crossover*, or *mutation* operators to be applicable for TSP. For example Nagata and Kobayashi (2013), developed GA for TSP where a new *crossover* called EAX-crossover is introduced. Furthermore, a pheromone-based crossover operator with a local-search operator work as a *mutation* operator to solve TSP (Zhao et al., 2008). An efficient algorithm to reduce the computation time of GA is proposed in Tsai et al. (2014). Another improved GA is also proposed for TSP in which a new mutation operator is produced by Thanh et al. (2015). There are many other GA-based TSP have been also proposed as surveyed in Vaishnav et al. (2017).

In this paper, a new local search mechanism based on three neighbourhood structure (*inverse*, *insert*, and *swap*) along with 2-*opt* neighbourhood procedure is proposed to generate the initial population for GA. Note that one of the main obstacles for GA is building its initial population. When GA initiated with a strong initial population, the convergence rate and the diversity aspect will be more stronger. In order to validate the performance of the proposed algorithm, three TSP datasets (i.e., TSPLIB, national TSP, and VLSI TSP) of different complexities and sizes, are used including 39 TSP instances. The size range of the TSP instances are from 150 to 33,810 cities. For evaluation purposes, the impact of each neighbouring operator on the performance of the proposed algorithm is studied. Consequently, the GA with the three neighbouring operators achieved the best results. For comparative evaluation, the results obtained by our proposed method is compared with those obtained by six well-regard methods using the same TSP instances. The proposed method is able to outperform other comparative methods in almost all TSP instances used.

The remaining parts of this paper is organised as follows: The definition and formulation of TSP problem are discussed and the fundamental GA is also overviewed in Section 2. The proposed method with the new neighbouring operators are discussed in Section 3. The experimental results and discussions are analysed in Section 4. Finally, the conclusion and possible future research is discussed in Section 5.

2 Research background

In order to make a self-exploratory paper, the problem of TSP is initially modeled in terms of optimisation problem in Subsection 2.1. Also, the GA and its basic form is also discussed as the solver for TSP problem, suggested in the present work (please see Subsection 2.2).

2.1 Travelling salesman problem

The TSP is one of the most popular NP-hard combinatorial optimisation problems used to find the shortest closed tour that visits each city one and only one. TSP traditionally represented as a bi-directed graph $G = (V, A)$, where V is set of vertices (i.e., cities), and A is a set or arcs (i.e., the connections between cities). The cost matrix D of size $V \times V$ is used to store the distance between all pairs of cities, where each element d_{ij} represents the distance between the two cities v_i, and v_j. Generally, the cost matrix can be classified to symmetric or asymmetric. In the symmetric case, the distance between cities are independent of the direction of traversing the arcs ($d_{ij} = d_{ji}$), whereas, $d_{ij} \neq d_{ji}$ in the asymmetric case. a_{ij} is one if the arc between the two cities v_i and v_j in the tour, and zero otherwise.

Mathematically, the objective function of the TSP can be formulated as follows:

$$z = \min \sum_{i} \sum_{j} d_{ij}.a_{ij} \ , \tag{1}$$

where z is the total closed tour length; d_{ij} is the distance between the two cities v_i and v_j; a_{ij} is the existence of the arc between the two cities in the tour.

2.2 Genetic algorithm

GA was developed by Holland (1975), to mimic the natural phenomenon of Darwin Evolution Theory and based on a well-known principle in evolution called '*survival of the fittest*'. GA starts with many solutions for many combinatorial optimisation problems, each of which is a vector of decision variables and each decision variable has a specific range of values (Goldberg, 1989; Holland, 1992). In the context of evolution, the set of solutions is equivalent to *population*, each solution is analogous to *chromosome*, each decision variable to *gene*, and each value of the decision variables to *allele*.

In order to apply a successful GA to any optimisation problem, the objective function and problem representation have to be first properly adjusted together with parameter tuning. Typically, GA has a set of parameters including the size of the population (P_s), the number of generation (P_n), the crossover rate (P_c), and the mutation rate (P_m). In order to build an efficient and robust GA, the parameters settings to each optimisation problem have to be studied well.

Algorithm 1 shows the high-level schematic pseudo-code of GA, which starts with a population of candidate solutions $\mathbf{X}$, where $\mathbf{X}$ is an augmented matrix of size $P_s \times N$ and N is the number of decision variables in each solution. Initially, the population $\mathbf{X}$ is filled with random candidate solutions across the problem search space, e.g., $\mathbf{X} = \{\boldsymbol{x}^1, \boldsymbol{x}^2, \ldots, \boldsymbol{x}^{P_s}\}$. Each candidate solution $\boldsymbol{x}^i$, is evaluated based on an objective function. The improvement loop in GA (see Algorithm 1, Line 3 to Line 9), repeats the following steps until a termination criterion is met: selecting the parents (new population $\mathbf{X}'$) that will be used to generate the next population, which will pairwise crossover with a probability of P_c to come up with a new population $\mathbf{X}''$. Afterwards, each pairwise solution will be checked for whether it should be mutated with probability P_m to come up with $\mathbf{X}'''$, the new population will again be evaluated and the $\mathbf{X}'''$ will be substituted with the

population **X** based on such selection method. This is done to filter the offsprings as fit or not. This process will be repeated several times trying to reach an optimal solution.

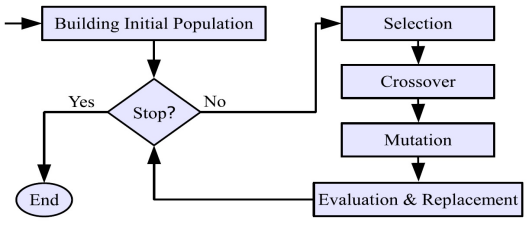
Algorithm 1 Genetic algorithm

```
X ← Generate_Initital_Population
Evaluate(X)
while (Stopping Criterion is not met) do
    X′ ← Selection(X)
    X″ ← Crossover (X′)
    X‴ ← Mutation (X″)
    Evaluate(X‴)
    X ← Replacement(X‴ ∪ X)
end while
```

3 Proposed MGA-TSP algorithm

In this section, the proposed modernised genetic algorithm for solving the TSP problem (MGA-TSP), is discussed in details. The main focus of this paper is on the way of generating the initial solution for GA. The initial population is build based on the three proposed neighbouring operators discussed in the following sections as well as the original 2-*opt* neighbouring procedure. The main motivation behind this idea is to provide a very good individuals for GA to enhance its capability in finding the global optima. The flowchart of the proposed method is given in Figure 1. The proposed method is initiated by a set of individuals produced by $\mathcal{N}$Opt of MGA-TSP operators. Generation after generation, proportionable selection, EAX crossover, simple mutation and greedy replacement operators are used in the improvement loop. The proposed algorithm is terminated after a set of generations determined by generation number. The output of the proposed method is an optimal tour for TSP. The following subsection will thoroughly discuss each step of the proposed method.

Figure 1 Flowchart of GA algorithm (see online version for colours)



3.1 Initialise TSP and GA parameters

For TSP problem, each individual in GA algorithm is represented as a chromosome $\boldsymbol{x} = (x_1, x_2, \ldots, x_N)$ of N cities. The value range of each gene indexed by x_i is the city label $x_i \in (1, 2, \ldots, N)$. Note that the cities in each individual are permutated. This means that each city will appear in each individual one and only one time. This solution representation is known as path representation. Each individual will be evaluated using the fitness function formulated in equation (1).

For GA, its parameters will be also initialised in this step such as crossover rate (P_c) and mutation rate (P_m). the proportional selection scheme is used to utilise the survival of the fittest principle of natural selection. Also the neighbouring procedures in GA will be used based on $\mathcal{N}_r$ rate, which reflect the usage rate of neighbouring function $\mathcal{N}_i$.

3.2 Building initial population

The focal point of this paper is the process of how to build the initial population for TSP. The initial population is normally built based on a specific random mechanism as many as population size. As conventionally known, the search space compromise the set of possible solutions. In TSP case, the search space of N citeps has $N!$ possible solutions. It is worth mentioning that the success of the GA often depends on the quality and the distribution of the initial population.

Algorithm 2 Generate initial population

```
for i = 1 to Pop_size do
    if rand < N_r then
        x_i = N_1_2-opt()      ▷ { See Algorithm 3.}
    else
        x_i1 = N_2_Inverse()
        x_i2 = N_3_Insert(x_i1)
        x_i3 = N_4_Swap(x_i2)
        x_i = Short_tour(x_i1, x_i2, x_i3)
    end if
end for
```

In our proposed method, the initial population for TSP, is generated using five different neighbourhood operators, The process of how these operators work are based on the neighbouring rate $\mathcal{N}_r$. Algorithm 2 shows how these operators collaborate to generate the initial population. The process of each neighbouring operator is next discussed.

$\mathcal{N}1$ – 2-opt operator

This algorithm is widely used in the domain of TSP initially established in Johnson and McGeoch (1997). It is normally used in the process of generating the initial population of TSP with GA. As pseudocoded in Nagata and Kobayashi (2013), the 2-*opt* operator is provided in Algorithm 3. It generates each individual using local search strategy in which the possible move in 2-*opt* can be formulated as a 4-tuple of vertices. In 2-*opt* for instance, the edge $E_{1,2} = (v_1, v_2)$ and the edge $E_{3,4} = (v_3, v_4)$ will be removed and reconstructed as the edge $E_{1,3} = (v_1, v_3)$ and the edge $E_{2,4} = (v_2, v_4)$. In practice, Algorithm 3 established in Nagata and Kobayashi (2013), illustrates how the 2-*opt* works. It works as a local search with best improvement strategies on several trials of its neighbouring vertices.

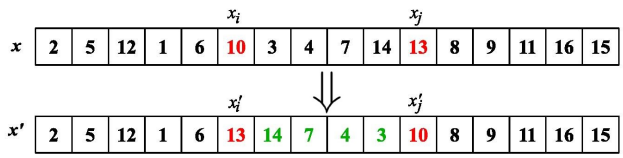
Algorithm 3 Procedure local-search() or 2-opt()

```
1: Randomly generate a solution H := {1,...,N};
2: repeat
3:    Randomly select v1 ∈ H;
4:    for i:= 0 to 1 do
5:       v2 := Neighbour[v1][i];
6:       for j:= 1 to trial do
7:          v3 := near[v1][j];      ▷ {trial < N}
8:          if -d(v1, v2) + d(v1, v3)≥ 0 then break;
9:          v4 := Neighbour[v3][(i + 1) mod 2];
10:         if (−d(v1, v2) − d(v3, v4) + d(v1, v3) + d(v2, v4) < 0)
            then
11:            Update the current tour and H. Go to line 3;
12:         end if
13:      end for
14:   end for
15: until becomes empty
```

$\mathcal{N}2$ – *inverse operator*

The function inverse $(\mathbf{x}, i, j)$ will inverse the cities between the index i and index j as follows: $\boldsymbol{x}' = (\ldots, x'_i = x_j, x'_{i+1} = x_{j-1}, \ldots, x'_{i+k} = x_{j-k}, \ldots, x'_j = x_i, \ldots)$. A visual representation of inverse process is provided in Figure 2. Two edges are exchanged by inverse operator for TSP.

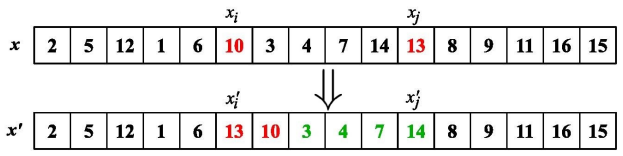
Figure 2 Inverse neighbourhood structure (see online version for colours)



$\mathcal{N}3$ – *insert operator*

The function insert$(\mathbf{x}, i, j)$ will shift the city in index j to index i consecutively as follows: $\boldsymbol{x}' = (\ldots, x'_i = x_j, x'_{i+1} = x_i, x'_{i+2} = x_{i+1}, \ldots, x'_j = x_{j-1}, \ldots)$. A visual representation of insert process is provided in Figure 3.

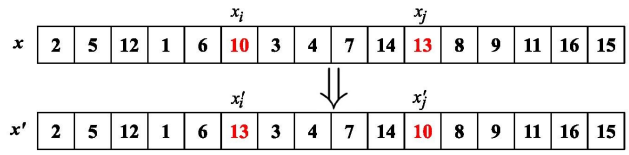
Figure 3 Insert neighbourhood structure (see online version for colours)



$\mathcal{N}4$ – *swap operator*

The function swap $(\mathbf{x}, i, j)$ will swap the two cities in the index i and index j as follows: $x'_i = x_j$ and $x'_j = x_i$. A visual representation of swap process is provided in Figure 4.

Figure 4 Swap neighbourhood structure (see online version for colours)



3.3 *Selection operator*

In this step, the fittest solutions are selected from the population to generate the next generation. There are several selection mechanism. In this paper, the proportional selection method established in the original GA (Holland, 1975) is used. In proportional selection method, the fittest individuals in the population have higher chances to be selected to generate the next population than the other individuals.

3.4 *Crossover operator (EAX)*

EAX is an efficient TSP-based crossover operator widely used by several GA-based TSP studies (Nagata and Kobayashi, 2013; Sanches et al., 2017a, 2017b). Simply EAX initially combines two parents stored in a population randomly selected into a single graph G. These two parent graphs are merged together. Absolutely, there are several redundant edges should be modified. Therefore, the AB-cycles in the merged graph are defined in Sanches et al. (2017a), as: one edge is taken from first parent graph (say A); then another edge is taken from the second parent graph (say B) alternatively and continually (a-b-a-b-a-…-a-b). Note that each graph G can be splitted into set of AB-cycles. In AB-cycle, the E-set is the number of AB-cycles generated, which will be used to cut one parent into subcircuits. These subcircuits are then combined using greedy criteria depend on the Hamiltonian cycle aspects in which the circuit of the shorter edge and not in both parents (A and B) will be considered. Further discussion about EAX can be found in Nagata and Kobayashi (2013) and Sanches et al. (2017a).

3.5 *Mutation operator*

The mutation operator is normally worked after the crossover operator to preserve the diversity during the search. This operator will improve the ability of GA to find the optimal solution. The main benefit of this operator is that it enables the GA to avoid the premature convergence situation (Holland, 1975; Alipour et al., 2017). For TSP, the simple mutation operator is used where the positions of two cities are selected and their position will be swapped. Normally, this operator is controlled by the mutation rate (P_m), which is used with a small value.

3.6 Evaluation and replacement operators

The proposed MGA-TSP algorithm calculate the objective function for each individual. The fittest generated individuals will replace the others in the population, if better.

3.7 Termination condition

The maximum number of iterations is used as a termination criteria for the proposed GA-based algorithm. The output of the the proposed method will be the individual with the best lowest objective function value.

4 Experimental results and discussion

In this section, a full description is reported for the experimental and tested results that are obtained using the proposed MGA-TSP algorithm. A comparison study is presented to show the superior of our proposed algorithm, which proves its outperformance against the obtained results of other existing TSP algorithms in the literature. Basically, tested dataset and evaluation metrics are fully explained in the next subsection, before discussing the experimental results.

The datasets for testing our proposed algorithms, are carefully selected from three well-known benchmark sets, which are TSP:TSPLIB, national TSP, and VLSI TSP, as seen in Table 1. These datasets are different in terms of number of cities and complexity. The proposed MGA-TSP algorithm was tested on 39 instances that are chosen from the main benchmark datasets.

4.1 Effect the Neighbouring operators on performance of proposed MGA-TSP method

In this section, the effect of the employed three neighbouring operators (*inverse*, *insert*, *swap*), on the performance of proposed MGA-TSP method, are thoroughly studied. Due to the fact that the initial population is strongly generated based on these three proposed neighbouring operators along with 2-*opt* neighbouring operator. The validity of these three operators are tested using eight versions of proposed method abbreviated in Table 2. These eight variations represents the all possible combinations of the GA-based TSP with the three neighbouring operators.

Each variation of the proposed method is experimented with all TSP instances. The results obtained by each variation are summarised in Table 3. The results are summarised in terms of 10 runs executed for each TSP instance. Note that the 'OPT (BKS)' column refers to the known optimal solution for each TSP instance. The best results from each variant for each TSP instance are highlited in ital font. It is worthy mentioning that the result of the first 18 TSP instances are omitted from the table because all variations of the proposed method are able to obtain the best known results. Therefore the table recorders results of the other 22 more complicated TSP instances.

Table 1 The datasets of used TSP instances

Instance number	*Instance name*	*General description/ comment*
1	Ch150	150 city problem (churritz)
2	Kroa150	150-city problem (Krolak/Felts/Nelson)
3	Krob150	150-city problem B (Krolak/Felts/Nelson)
4	Pr152	152-city problem (Padberg/Rinaldi)
5	U159	Drilling problem (Reinelt)
6	Rat195	Rattled grid (Pulleyblank)
7	D198	Drilling problem (Reinelt)
8	Kroa200	200 city problem A (Krolak/Felts/Nelson)
9	Krob200	200 city problem B (Krolak/Felts/Nelson)
10	Ts225	225 city problem (Juenger)
11	Pr226	226 city problem (Padberg/Rinaldi)
12	Gil262	262 city problem (Gillet/Johnson)
13	Pr264	264 city problem (Padberg/Rinaldi)
14	Pr299	299 city problem (Padberg/Rinaldi)
15	Lin318	318 city problem (Lin/Kernighan)
16	Rd400	400 city random TSP(Reinelt)
17	Fl417	Drilling problem (Reinelt)
18	Pr439	439 city problem (Padberg/Rinaldi)
19	Pcb442	Drilling problem (Groetschel)
20	U574	Drilling problem (Reinelt)
21	Rat575	Rattled grid (Pulleyblank)
22	U724	Drilling problem (Reinelt)
23	Rat783	Rattled grid (Pulleyblank)
24	Pr1002	1,002 city problem (Padberg/Rinaldi)
25	Pcb1173	Drilling problem (Juenger/Reinelt)
26	D1291	Drilling problem (Reinelt)
27	Rl1323	1,323 city TSP (Reinelt)
28	Fl1400	Drilling problem (Reinelt)
29	D1655	Drilling problem (Reinelt)
30	Vm1748	1,784 city problem (Reinelt)
31	U2319	Drilling problem (Reinelt)
32	Pcb3038	Drilling problem (Junger/Reinelt)
33	Fnl4461	Die 5 neuen Laender Deutschlands (ExDDR) (Bachem/Wottawa)
34	Rl5934	5,934 city TSP (Reinelt)
35	Pla7397	Programmed logic array (Johnson)
36	Usa13509	Cities with pop. at least 500 in the continental US
37	Brd14051	BR Deutschland in den Grenzen von 1989 (Bachem/Wottawa)
38	D18512	Bundesrepublik Deutschland (Bachem)
39	Pla33810	Programmed logic array (Johnson)

Notes: TSP: TSPLIB url: http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsp/.
National TSP url: http://www.math.uwaterloo.ca/tsp/world/countries.html
VLSI datasets url: http://www.math.uwaterloo.ca/tsp/vlsi/summary.html

Table 2 The variations of the proposed MGA-TSP methods using three neighbouring operators (*inverse*, *insert*, *swap*)

Key	*The proposed MGA-TSP variations*
M1	MGA-GA with inverse, insert, swap optrs.
M2	MGA-GA without inverse, insert, swap optrs.
M3	MGA-GA with inverse operator only
M4	MGA-GA with insert operator only
M5	MGA-GA with swap operator only
M6	MGA-GA with inverse and insert operators
M7	MGA-GA with inverse and swap operators
M8	MGA-GA with insert and swap operators

Table 3 The results of the eight variations of our proposed MGA-TSP method for 22 TSP instances

S/N	*Instance*	*OPT(BKS)*	*M1*	*M2*	*M3*	*M4*
18	Pr439	107,217.00	*107,217.00*	107,218.50	107,217.50	107,218.00
19	Pcb442	50,778.00	*50,778.00*	*50,778.00*	*50,778.00*	*50,778.00*
20	U574	36,905.00	*36,905.00*	*36,905.00*	*36,905.00*	*36,905.00*
21	Rat575	6,773.00	*6,773.00*	6,773.10	*6,773.00*	*6,773.00*
22	U724	41,910.00	*41,910.00*	41,911.20	41,910.60	41,910.60
23	Rat783	8,806.00	*8,806.00*	*8,806.00*	*8,806.00*	*8,806.00*
24	Pr1002	259,045.00	*259,045.00*	*259,045.00*	*259,045.00*	259,046.70
25	Pcb1173	56,892.00	*56,892.00*	56,892.50	56,892.10	56,893.10
26	D1291	50,801.00	*50,801.00*	*50,801.00*	*50,801.00*	*50,801.00*
27	Rl1323	270,199.00	*270,199.00*	270,201.70	270,229.90	270,203.50
28	Fl1400	20,127.00	20,130.70	20,134.40	20,140.40	20,139.80
29	D1655	62,128.00	62,129.30	62,132.30	*62,129.20*	62,131.40
30	Vm1748	336,556.00	*336,556.00*	336,558.40	*336,556.00*	336,559.20
31	U2319	234,256.00	*234,346.80*	234,363.20	234,361.50	234,363.20
32	Pcb3038	137,694.00	*137,694.00*	*137,694.00*	137,695.00	137,695.00
33	Fnl4461	182,566.00	182,568.70	182,569.50	182,571.40	*182,568.30*
34	Rl5934	556,045.00	*556,082.10*	556,295.80	556,187.10	556,239.30
35	Pla7397	23,260,728.00	*23,262,643.40*	23,263,644.80	23,263,180.10	23,264,915.60
36	Usa13509	19,982,859.00	19,983,654.00	*19,983,593.80*	19,983,454.70	19,984,011.60
37	Brd14051	469,385.00	*469,391.30*	469,393.90	469,393.10	469,391.60
38	D18512	645,238.00	*645,248.40*	645,248.50	645,251.00	645,252.20
39	Pla33810	66,048,945.00	*66,067,277.50*	66,068,408.70	66,070,564.20	66,068,696.40
S/N	*Instance*	*OPT(BKS)*	*M5*	*M6*	*M7*	*M8*
18	Pr439	107,217.00	107,218.00	107,217.50	107,219.50	107,218.00
19	Pcb442	50,778.00	*50,778.00*	*50,778.00*	*50,778.00*	*50,778.00*
20	U574	36,905.00	*36,905.00*	*36,905.00*	*36,905.00*	*36,905.00*
21	Rat575	6,773.00	6,773.10	*6,773.00*	*6,773.00*	*6,773.10*
22	U724	41,910.00	41,910.60	41,910.60	41,911.20	41,910.60
23	Rat783	8,806.00	*8,806.00*	*8,806.00*	*8,806.00*	*8,806.00*
24	Pr1002	259,045.00	259,049.70	*259,045.00*	*259,045.00*	*259,045.00*
25	Pcb1173	56,892.00	56,892.10	56,892.10	*56,892.00*	56,892.20
26	D1291	50,801.00	50,805.10	*50,801.00*	*50,801.00*	*50,801.00*
27	Rl1323	270,199.00	270,201.70	270,200.50	*270,199.00*	270,203.20
28	Fl1400	20,127.00	20,130.80	20,138.20	20,142.30	20,134.90
29	D1655	62,128.00	62,130.80	62,131.00	62,233.10	62,183.60
30	Vm1748	336,556.00	336,557.20	336,567.50	*336,556.00*	*336,556.00*
31	U2319	234,256.00	*234,346.80*	234,355.00	*234,355.00*	*234,355.00*
32	Pcb3038	137,694.00	*137,694.00*	137,694.50	137,695.70	*137,694.00*
33	Fnl4461	182,566.00	182,570.70	182,571.20	182,570.10	182,570.40
34	Rl5934	556,045.00	556,176.00	556,207.30	556,169.50	556,229.60
35	Pla7397	23,260,728.00	23,264,689.60	23,264,546.00	23,265,510.60	23,263,946.20
36	Usa13509	19,982,859.00	19,984,049.30	19,983,685.10	19,983,902.30	19,983,955.90
37	Brd14051	469,385.00	469,391.90	469,393.80	469,393.50	469,395.20
38	D18512	645,238.00	645,250.60	645,251.10	*645,248.40*	645,250.60
39	Pla33810	66,048,945.00	66,069,872.00	66,070,847.80	66,068,291.00	66,068,696.40

Table 4 The abbreviations of the comparative methods

Abbrev. and refs.	*The comparative method*
1 SOS-SA (Ezugwu et al., 2017)	Simulated annealing-based symbiotic organisms search optimisation alg.
2 LBSA (Zhan et al., 2016)	List-based simulated annealing algorithm
3 MSA-IBS (Wang et al., 2015)	Multiagent simulated annealing alg. with instance-based sampling
4 EAX (Nagata and Kobayashi, 2013)	Genetic algorithm using edge assembly crossover
5 GA-PSO-ACO (Deng et al., 2012)	Two-stage hybrid swarm intelligence optimisation algorithm
6 GA-PSO-ACO (Geng et al., 2011)	Adaptive simulated annealing algorithm with greedy search

As borne out by the results recorded in Table 3, almost all best results are obtained by M1 variation, which the proposed method combined with the proposed three neighbouring operators. Some best results are obtained by M7 and M8, which combined only two neighbouring operators. Notably, almost no best results can be achieved by M3, M4, and M5 with only one neighbouring operator. Apparently, although the combination of the three neighbouring operators seems efficient for the proposed GA-based TSP, the most effective neighbouring operator is Swap. This is because most of the best results achieved when the swap neighbouring operator is exist. In a nutshell, combining the three neighbouring operators with 2-*opt* as a local search operator to build the initial population for GA-based TSP seems to have a direct improve to the final outcomes.

Table 5 The comparative results of the proposed method against six comparative methods

S/N	Instance	OPT (BKS)	Proposed MGA-TSP	SOS-SA (2017)	LBSA (2016)	MSA-IBS (2015)	EAX (2013)	GA-PSO-ACO (2012)	ASA-GS (2011)
			Mean	Mean	Mean	Mean	Mean	Mean	Mean
1	Ch150	6,528.00	*6,528.00*	6,529.84	6,529.80	6,529.00	-NA-	-NA-	6,529.83
2	Kroa150	26,524.00	*26,524.00*	26,524.02	*26,524.00*	*26,524.00*	-NA-	26,803.00	26,538.60
3	Krob150	26,130.00	*26,130.00*	26,131.83	26,137.00	26,135.00	-NA-	-NA-	26,178.10
4	Pr152	73,682.00	*73,682.00*	73,682.18	*73,682.00*	*73,682.00*	73,695.60	73,989.00	73,694.70
5	U159	42,080.00	*42,080.00*	42,080.98	*42,080.00*	*42,080.00*	*42,080.00*	42,506.00	42,398.90
6	Rat195	2,323.00	*2,323.00*	2,326.60	2,328.00	2,330.20	*2,323.00*	2,362.00	2,348.05
7	D198	15,780.00	*15,780.00*	15,782.11	*15,780.00*	*15,780.00*	*15,780.00*	-NA-	15,845.40
8	Kroa200	29,368.00	*29,368.00*	29,370.78	29,373.80	*29,378.00*	-NA-	-NA-	29,438.40
9	Krob200	29,437.00	*29,437.00*	29,449.82	29,442.20	29,439.80	-NA-	-NA-	29,513.10
10	Ts225	126,643.00	*126,643.00*	126,701.09	*126,643.00*	*126,643.00*	*126,643.00*	-NA-	126,646.00
11	Pr226	80,369.00	*80,369.00*	80,369.31	*80,369.80*	*80,369.00*	*80,369.00*	-NA-	80,687.40
12	Gil262	2,378.00	*2,378.00*	2,381.92	2,379.20	2,378.80	*2,378.00*	2,439.00	2,398.61
13	Pr264	49,135.00	*49,135.00*	49,135.72	*49,135.00*	*49,135.00*	*49,135.00*	-NA-	49,138.90
14	Pr299	48,191.00	*48,191.00*	48,227.93	48,221.20	48,226.40	*48,191.00*	48,763.00	48,326.40
15	Lin318	42,029.00	*42,029.00*	42,179.31	42,195.60	42,184.40	-NA-	42,771.00	42,383.70
16	Rd400	15,281.00	*15,281.00*	15,451.81	15,350.40	15,429.80	*15,281.00*	15,503.00	15,429.80
17	Fl417	11,861.00	*11,861.00*	11,877.52	11,867.80	11,875.60	*11,861.00*	-NA-	12,043.80
18	Pr439	107,217.00	*107,217.00*	107,561.15	107,465.20	107,407.20	107,217.50	-NA-	110,226.00
19	Pcb442	50,778.00	*50,778.00*	50,871.82	50,877.00	50,970.00	*50,778.00*	51,494.00	51,269.20
20	U574	36,905.00	*36,905.00*	37,164.49	37,164.60	37,155.80	*36,905.00*	-NA-	37,369.80
21	Rat575	6,773.00	*6,773.00*	6,839.52	6,837.40	6,839.80	*6,773.00*	6,952.00	6,904.82
22	U724	41,910.00	*41,910.00*	42,262.11	42,252.00	42,212.20	*41,910.00*	42,713.00	42,470.40
23	Rat783	8,806.00	*8,806.00*	8,899.55	8,888.20	8,893.40	*8,806.00*	9,126.00	8,982.19
24	Pr1002	259,045.00	*259,045.00*	261,802.49	261,805.20	261,481.80	*259,045.00*	266,774.00	264,274.00
25	Pcb1173	56,892.00	*56,892.00*	57,569.94	57,431.80	57,561.60	56,893.10	-NA-	57,820.50
26	D1291	50,801.00	*50,801.00*	51,291.09	51,198.80	51,343.80	*50,801.00*	52,443.00	52,252.30
27	Rl1323	270,199.00	*270,199.00*	271,710.63	271,714.40	271,818.40	*270,199.00*	-NA-	273,4 44.00
28	Fl1400	20,127.00	20,130.70	20,231.02	20,249.40	20,374.80	*20,127.00*	-NA-	20,782.20
29	D1655	62,128.00	*62,129.30*	64,111.92	63,001.40	62,893.00	62,131.20	65,241.00	64,155.90
30	Vm1748	336,556.00	*336,556.00*	336,719.39	339,710.80	339,617.80	336,572.60	-NA-	343,911.00
31	U2319	234,256.00	*234,346.80*	235,338.09	235,975.00	235,236.00	234,371.40	-NA-	236,744.00
32	Pcb3038	137,694.00	*137,694.00*	139,701.81	139,635.20	139,706.20	*137,694.00*	-NA-	141,242.00
33	Fnl4461	182,566.00	*182,568.70*	185,546.04	185,509.40	185,535.40	182,570.70	192,574.00	187,409.00
34	Rl5934	556,045.00	*556,082.10*	566,211.72	566,053.00	566,166.80	556,174.00	-NA-	575,437.00
35	Pla7397	23,260,728.00	*23,262,643.40*	23,800,000.00	23,800,000.00	23,800,000.00	23,264,735.70	-NA-	24,166,453.00
36	Usa13509	19,982,859.00	*19,983,654.00*	21,400,000.00	20,400,000.00	20,400,000.00	19,983,837.10	-NA-	20,811,106.00
37	Brd14051	469,385.00	*469,391.30*	478,098.91	478,010.00	478,609.60	469,398.10	503,560.00	486,197.00
38	D18512	645,238.00	645,248.40	659,457.45	657,457.20	658,149.20	*645,248.00*	-NA-	669,445.00
39	Pla33810	66,048,945.00	*66,067,277.50*	68,076,220.23	68,029,226.40	68,075,607.00	66,068,266.10	72,420,147.00	69,533,166.00

4.2 Comparative evaluation

In order to validate the performance of the proposed method, the best means of the results obtained are compared with other six well-regard methods summarised and abbreviated in Table 4. These methods produced the best known results recorded previously for the TSP instances used. Some of these method used GA as a primary method to solve TSP and others used hybrid technique designed especially for TSP which are published recently.

The key comparative results are recorded in the Table 5. These results obtained by the proposed method, are compared with those obtained by the other comparative methods. Numbers in Table 5, shows the mean results of 10 runs executed by each comparative method for all TSP instances. The best results (lowest is best) is highlighted in ital font. The mark '-NA-' in the table refers to the corresponding method that did not experimented with such TSP instance or it cannot solve it. Again, the best known result for each TSP instances is recorded in the third column [i.e., OPT (BKS) column]. The TSP instance name has the

number of cities to be visited which appeared in the second column.

As shown in Table 5, the performance of the proposed method is better than other comparative methods in 15 datasets. Furthermore, the proposed method is obtained the best published results in 22 datasets as achieved by other comparative six methods. However, the proposed method is get the second best results for '*D18512*' instance, where the best results for this instance is obtained by EAX algorithm. It should be noted that the EAX algorithm is ranked the second, where it obtained the best results on 25 out of 39 datasets.

5 Conclusions and future work

In this paper, a new local search strategy based on efficient neighbouring operators, is used to build a strong initial population for GA to solve TSP problem. The proposed new operator is called MGA-TSP, which is used three neighbouring operators (*inverse*, *insert*, and *swap*). Each one has its ability to navigate the TSP search space in a different manner and thus maintain the diversity level of GA algorithm. These three neighbouring operators are cooperated with 2-*opt* strategy to build the initial population. Thereafter, GA will iteratively improves that population using proportional *selection*, EAX *crossover* and simple *mutation* operators until the optimal solution is reached.

For evaluation purposes, three popular TSP datasets including 39 problem instances are used, which are TSPLIB, National TSP, and VLSI TSP. These datasets are different in terms of problem size and complexity. The effect of each neighbouring operators and all possible combinations are investigated in the form of convergence scenarios. This is to study the best configuration of the proposed local search method in the performance of GA algorithm. In a nutshell, the GA with the three neighbouring operators in the local search, building the initial solution for GA achieved the best outcomes with a reasonable complexity. Furthermore, the outcomes of the proposed method are compared with those produced by six other well-established methods using the same datasets. Interestingly, the proposed method is able to excel the other comparative methods for almost all TSP problem instances.

Building a strong initial solution for TSP with GA has direct impact on the final outcomes. Therefore, other neighbouring methods that are adapted for TSP can be utilised and further studied to improve the GA connectivity to the TSP search space by means of navigating the inaccessible regions and thus improve the final results. Furthermore, the effect of the GA parameters such as mutation rate and crossover rate on the convergence behaviour of MGA-TSP should be also studied in the future to promote the best GA parameter setting. Other challenging TSP datasets can be also investigated to further ensure the validity of the proposed MGA-TSP in the future work.

Acknowledgments

Thanks are due to the anonymous referees for their insightful comments and marvelous efforts.

References

Albayrak, M. and Allahverdi, N. (2011) 'Development a new mutation operator to solve the traveling salesman problem by aid of genetic algorithms', *Expert Systems with Applications*, Vol. 38, No. 3, pp.1313–1320.

Alijla, B.O., Wong, L-P., Lim, C.P., Khader, A.T. and Al-Betar, M.A. (2014) 'A modified intelligent water drops algorithm and its application to optimization problems', *Expert Systems with Applications*, Vol. 41, No. 15, pp.6555–6569.

Alipour, M.M., Razavi, S.N., Derakhshi, M.R.F. and Balafar, M.A. (2017) 'A hybrid algorithm using a genetic algorithm and multiagent reinforcement learning heuristic to solve the traveling salesman problem', *Neural Computing and Applications*, pp.1–17.

Anantathanavit, M. and Munlin, M. (2016) 'Using k-means radius particle swarm optimization for the travelling salesman problem', *IETE Technical Review*, Vol. 33, No. 2, pp.172–180.

Applegate, D.L., Bixby, R.E., Chvatal, V. and Cook, W.J. (2011) *The Traveling Salesman Problem: A Computational Study*, Princeton Series in Applied Mathematics, 2nd ed., Princeton Univ. Press, Princeton, NJ.

Basu, S., Sharma, M. and Ghosh, P.S. (2017) 'Efficient preprocessing methods for Tabu search: an application on asymmetric travelling salesman problem', *INFOR: Information Systems and Operational Research*, Vol. 55, No. 2, pp.134–158.

Bayram, H. and Şahin, R. (2013) 'A new simulated annealing approach for travelling salesman problem', *Mathematical and Computational Applications*, Vol. 18, No. 3, pp.313–322.

Changdar, C., Mahapatra, G. and Pal, R.K. (2014) 'An efficient genetic algorithm for multi-objective solid travelling salesman problem under fuzziness', *Swarm and Evolutionary Computation*, Vol. 15, pp.27–37.

Chen, L., Sun, H-Y. and Wang, S. (2012) 'A parallel ant colony algorithm on massively parallel processors and its convergence analysis for the travelling salesman problem', *Information Sciences*, Vol. 199, pp.31–42.

Chen, S-M. and Chien, C-Y. (2011) 'Solving the traveling salesman problem based on the genetic simulated annealing ant colony system with particle swarm optimization techniques', *Expert Systems with Applications*, Vol. 38, No. 12, pp.14439–14450.

Deng, W., Chen, R., He, B., Liu, Y., Yin, L. and Guo, J. (2012) 'A novel two-stage hybrid swarm intelligence optimization algorithm and application', *Soft Computing*, Vol. 16, No. 10, pp.1707–1722.

Dorigo, M. and Gambardella, L.M. (1997) 'Ant colony system: a cooperative learning approach to the traveling salesman problem', *IEEE Transactions on Evolutionary Computation*, Vol. 1, No. 1, pp.53–66.

Ezugwu, A.E-S., Adewumi, A.O. and Frîncu, M.E. (2017) 'Simulated annealing based symbiotic organisms search optimization algorithm for traveling salesman problem', *Expert Systems with Applications*, Vol. 77, pp.189–210.

Geem, Z., Kim, J. and Loganathan, G. (2001) 'A new heuristic optimization algorithm: harmony search', *Simulation*, Vol. 76, No. 2, pp.60–68.

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

Geng, X., Chen, Z., Yang, W., Shi, D. and Zhao, K. (2011) 'Solving the traveling salesman problem based on an adaptive simulated annealing algorithm with greedy search', *Applied Soft Computing*, Vol. 11, No. 4, pp.3680–3689.

Goldberg, D.E. (1989) *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, Reading.

Helsgaun, K. (2000) 'An effective implementation of the lin-kernighan traveling salesman heuristic', *European Journal of Operational Research*, Vol. 126, No. 1, pp.106–130.

Holland, J.H. (1975) *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, MI, USA.

Holland, J.H. (1992) *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*, MIT Press, Cambridge, MA, USA.

Hoos, H.H. and Stützle, T. (2014) 'On the empirical scaling of run-time for finding optimal solutions to the travelling salesman problem', *European Journal of Operational Research*, Vol. 238, No. 1, pp.87–94.

Johnson, D.S. and McGeoch, L.A. (1997) 'The traveling salesman problem: a case study in local optimization', *Local Search in Combinatorial Optimization*, Vol. 1, pp.215–310.

Kalyani, R. (2015) 'Application of multi-core parallel programming to a combination of ant colony optimization and genetic algorithm', *Indian Journal of Science and Technology*, Vol. 8, No. S2, pp.138–142.

Kong, X., Sun, J. and Xu, W. (2006) 'Particle swarm algorithm for tasks scheduling in distributed heterogeneous system', in *ISDA'06, Sixth International Conference on Intelligent Systems Design and Applications*, IEEE, Vol. 2, pp.690–695.

Krause, J., Cordeiro, J., Parpinelli, R.S. and Lopes, H.S. (2013) 'A survey of swarm algorithms applied to discrete optimization problems', *Swarm Intelligence and Bio-inspired Computation: Theory and Applications*, pp.169–191, Elsevier Science & Technology Books, Oxford, pp.169–191.

Li, L., Cheng, Y., Tan, L. and Niu, B. (2012) *A Discrete Artificial Bee Colony Algorithm for TSP Problem*, pp.566–573, Springer Berlin, Heidelberg.

Lin, B.L., Sun, X. and Salous, S. (2016) 'Solving travelling salesman problem with an improved hybrid genetic algorithm', *Journal of Computer and Communications*, Vol. 4, No. 15, pp.98–106.

Liu, Y., Gao, C., Zhang, Z., Lu, Y., Chen, S., Liang, M. and Tao, L. (2017) 'Solving np-hard problems with physarum-based ant colony system', *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, Vol. 14, No. 1, pp.108–120.

Maity, S., Roy, A. and Maiti, M. (2016) 'An imprecise multi-objective genetic algorithm for uncertain constrained multi-objective solid travelling salesman problem', *Expert Systems With Applications*, Vol. 46, pp.196–223.

Matai, R., Singh, S. and Mittal, M.L. (2010) 'Traveling salesman problem: an overview of applications, formulations, and solution approaches', in *Traveling Salesman Problem, Theory and Applications*, InTech.

Matei, O. and Pop, P. (2010) 'An efficient genetic algorithm for solving the generalized traveling salesman problem', in *2010 IEEE International Conference on Intelligent Computer Communication and Processing (ICCP)*, IEEE, pp.87–92.

Michael, R.G. and David, S.J. (1979) *Computers and Intractability: A Guide to the Theory of np-Completeness*, pp.90–91, WH Free. Co., San Francisco.

Mladenović, N., Todosijević, R. and Urošević, D. (2013) 'An efficient general variable neighborhood search for large travelling salesman problem with time windows', *Yugoslav Journal of Operations Research*, Vol. 23, No. 1, pp.19–30.

Nagata, Y. and Kobayashi, S. (2013) 'A powerful genetic algorithm using edge assembly crossover for the traveling salesman problem', *INFORMS Journal on Computing*, Vol. 25, No. 2, pp.346–363.

Ouaarab, A., Ahiod, B. and Yang, X-S. (2014) 'Discrete cuckoo search algorithm for the travelling salesman problem', *Neural Computing and Applications*, Vol. 24, Nos. 7–8, pp.1659–1669.

Pourhassan, M. and Neumann, F. (2015) 'On the impact of local search operators and variable neighbourhood search for the generalized travelling salesperson problem', in *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, ACM, pp.465–472.

Saji, Y. and Riffi, M.E. (2016) 'A novel discrete bat algorithm for solving the travelling salesman problem', *Neural Computing and Applications*, Vol. 27, No. 7, pp.1853–1866.

Sanches, D., Whitley, D. and Tinós, R. (2017a) 'Building a better heuristic for the traveling salesman problem: combining edge assembly crossover and partition crossover', in *Proceedings of the Genetic and Evolutionary Computation Conference*, ACM, pp.329–336.

Sanches, D., Whitley, D. and Tinós, R. (2017b) 'Improving an exact solver for the traveling salesman problem using partition crossover', in *Proceedings of the Genetic and Evolutionary Computation Conference*, ACM, pp.337–344.

Senthilkumar, M. and Prasanna, S. (2014) 'A modified and efficient genetic algorithm to address a travelling salesman problem', *International Journal of Applied Engineering Research*, Vol. 9, No. 10, pp.1279–1288.

Starkey, A., Hagras, H., Shakya, S. and Owusu, G. (2016) 'A multi-objective genetic type-2 fuzzy logic based system for mobile field workforce area optimization', *Information Sciences*, Vol. 329, pp.390–411.

Thanh, P.D., Binh, H.T.T. and Lam, B.T. (2015) 'New mechanism of combination crossover operators in genetic algorithm for solving the traveling salesman problem', in *Knowledge and Systems Engineering*, pp.367–379, Springer.

Tsai, C-W., Tseng, S-P., Chiang, M-C., Yang, C-S. and Hong, T-P. (2014) 'A high-performance genetic algorithm: using traveling salesman problem as a case', *The Scientific World Journal*.

Vahdati, G., Ghouchani, S.Y. and Yaghoobi, M. (2010) 'A hybrid search algorithm with hopfield neural network and genetic algorithm for solving traveling salesman problem', in *2010 The 2nd International Conference on Computer and Automation Engineering (ICCAE)*, IEEE, Vol. 1, pp.435–439.

Vaishnav, P., Choudhary, N. and Jain, K. (2017) 'Traveling salesman problem using genetic algorithm: a survey', *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, Vol. 2, No. 3, pp.105–108.

Wang, C., Lin, J., Lin, M. and Zhong, Y. (2016) 'Evolutionary harmony search algorithm with metropolis acceptance criterion for travelling salesman problem', *International Journal of Wireless and Mobile Computing*, Vol. 10, No. 2, pp.166–173.

Wang, C., Lin, M., Zhong, Y. and Zhang, H. (2015) 'Solving travelling salesman problem using multiagent simulated annealing algorithm with instance-based sampling', *International Journal of Computing Science and Mathematics*, Vol. 6, No. 4, pp.336–353.

Yi, Y. and Fang, Q-S. (2010) 'The improved hybrid genetic algorithm for solving tsp based on handel-c', in *2010 3rd International Conference on Advanced Computer Theory and Engineering (ICACTE)*, IEEE, Vol. 3, pp.V3–330.

Yoon, J-W. and Cho, S-B. (2011) 'An efficient genetic algorithm with fuzzy c-means clustering for traveling salesman problem', in *2011 IEEE Congress on Evolutionary Computation (CEC)*, IEEE, pp.1452–1456.

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*, Vol. 2016, p.8.

Zhao, F., Dong, J., Li, S. and Yang, X. (2008) 'An improved genetic algorithm for the multiple traveling salesman problem', in *Control and Decision Conference, CCDC 2008*, Chinese, IEEE, pp.1935–1939.