IEEJ TRANSACTIONS ON ELECTRICAL AND ELECTRONIC ENGINEERING
IEEJ Trans 2016; **11**(S1): S142–S154
Published online in Wiley Online Library (wileyonlinelibrary.com). DOI:10.1002/tee.22247

Paper

Immune Algorithm Combined with Estimation of Distribution for Traveling Salesman Problem

Zhe Xu*, Non-member
Yirui Wang**, Non-member
Sheng Li***, Non-member
Yanting Liu*, Non-member
Yuki Todo****, Non-member
Shangce Gao*[a], Member

This paper describes an artificial immune algorithm (IA) combined with estimation of distribution algorithm (EDA), named IA-EDA, for the traveling salesman problem (TSP). Two components are incorporated in IA-EDA to further improve the performance of the conventional IA. First, aiming to strengthen the information exchange during different solutions, two kinds of EDAs involving univariate marginal distribution algorithm and population-based incremental learning are altered based on the permutation representation of TSP. It is expected that new promising candidate solutions can be sampled from the constructed probabilistic model of EDA. Second, a heuristic refinement local search operator is proposed to repair the infeasible solutions sampled by EDA. Therefore, IA-EDA can alleviate the deficiencies of the conventional IA and can find better solutions for TSP by well balancing the exploitation and exploration of the search. Experiments are conducted based on a number of benchmark instances with size up to 100 000 cities. Simulation results show that IA-EDA is effective for improving the performance of the conventional IA and can produce better or competitive solutions than other hybrid algorithms. © 2016 Institute of Electrical Engineers of Japan. Published by John Wiley & Sons, Inc.

Keywords: estimation of distribution algorithm, immune algorithm, probabilistic model, traveling salesman problem, hybridization

Received 11 December 2015; Revised 16 February 2016

1. Introduction

The traveling salesman problem (TSP) is a classical problem in combinatorial optimization, which has been extensively studied over the past few decades [1]. The objective of the generalized TSP is to find a minimum total cost Hamiltonian cycle. Formally, consider a graph $G = \{N, E\}$, where N is a set of nodes representing cities and E is a set of arcs connecting these nodes. The distance between city i and city j is denoted by $d(i, j)$. Therefore, a generalized TSP consists of finding a permutation π of cities in the graph G that minimizes the total tour length $D(\pi)$:

$$D(\pi) = \sum_{i=1}^{N-1} d(\pi_i, \pi_{i+1}) + d(\pi_N, \pi_1). \quad (1)$$

There are several practical uses of TSP, such as vehicle routing, drilling, logistics, transportation, planning, gene sequencing problems, etc. Over the years, TSP has been the testing ground for numerous techniques inspired from a variety of sources.

Traditionally, mathematically exact techniques had been employed to solve TSPs, such as linear programming [2], dynamic programming [3], and branch-and-cut algorithm [4]. Although exact algorithms can find optimal solutions for TSP, their performance is limited due to the scale of instances, i.e. only small-scale instances of TSP can be well solved by these exact algorithms.

Nowadays, attempts are being made to solve the optimization problems by using meta-heuristics, which are mostly nature-inspired, like genetic algorithms (GA), simulated annealing (SA), artificial neural networks (ANN), artificial immune algorithms (IA), particle swarm optimization (PSO), ant colony optimization (ACO), artificial bee colony (ABC), estimation of distribution algorithms (EDA), tabu search (TS), greedy randomized adaptive search procedure (GRASP), and so on. All these meta-heuristics can be applied to a wide variety of problems. Some algorithms give a better solution for some particular problems than others. However, there is no specific algorithm to achieve the best solution for all optimization problems [5]. Thus, more effective hybridization of meta-heuristics with other types of algorithms for solving complex problems is required, which can lead to more efficient behavior and greater flexibility in application. The growing interest in this scenario of hybrid meta-heuristics and some of the typical application to the variants of TSP are summarized in Table I.

It is worth emphasizing that Table I aims at giving some insights into the research community of the hybrid meta-heuristics, rather than giving a comprehensive survey or a taxonomy (readers may refer to Ref. [48] for an example) of it. It is observed from Table I that continuous research has been carried out to hybridize different algorithms to achieve high performance in solving TSPs. Experimental results in these research works consistently

[a] Correspondence to: Shangce Gao. E-mail: gaosc@eng.u-toyama.ac.jp

* Faculty of Engineering, University of Toyama, Toyama-shi, 930-8555 Japan
** College of Information Science and Technology, Donghua University, Shanghai, 201620 China
*** College of Computer Science and Technology, Taizhou University, Jiangsu, 225300 China
**** School of Electrical and Computer Engineering, Kanazawa University, Kakuma-chou, Kanazawa-shi, 920-1192 Japan

Table I. Summary of different hybrid meta-heuristics and their applications to variants of TSP

Researchers	Hybrid meta-heuristic	Application
Deng *et al*. [6]	GA + ACO + PSO	Generalized TSP
Wang *et al*. [7]	GA + ACO	Generalized TSP
Mavrovouniotis and Yang [8]	GA + ACO	Dynamic TSP
Baraglia *et al*. [9]	Compact GA + local search (Lin-Kernighan)	Generalized TSP
Tsai *et al*. [10]	Parallel GA + ACO	Generalized TSP
Nguyen *et al*. [11]	Steady-state GA + local search (Lin-Kernighan)	Generalized TSP
Xing *et al*. [12]	GA + local determinate/stochastic search + global optimization	Generalized TSP
Lin *et al*. [13]	GA + Newton–Raphson search	Generalized TSP
Dong *et al*. [14]	Cooperative GA + ACO	Generalized TSP
Abbattista *et al*. [15]	GA + ACO	Generalized TSP
Lope and Coelho [16]	GA + PSO	Blind TSP
Chen and Flann [17]	GA + SA	Generalized TSP
Creput and Koukam [18]	GA + ANN (self-organization map)	Generalized TSP
Jin *et al*. [19]	GA + ANN	Generalized TSP
Glover *et al*. [20]	GA + TS	Generalized TSP
Shim *et al*. [21]	GA + EDA	Multiobjective TSP
Marinakis *et al*. [22]	GA + GRASP	Generalized TSP
Chen and Chien [23]	GA + SA + ACO + PSO	Generalized TSP
Martin and Otto [24]	SA + local search (Lin-Kernighan)	generalized TSP
Malek *et al*. [25]	SA + TS	Generalized TSP
Geng *et al*. [26]	SA + greedy search	Generalized TSP
Pepper *et al*. [27]	SA + Monte Carlo method (demon algorithm)	Generalized TSP
Shim *et al*. [28]	SA + EDA + hill climbing + evolutionary gradient search	Multiobjective multiple TSP
Marinakis and Marinaki [29]	GRASP + PSO	probabilistic TSP
Marinakis and Marinaki [30]	GRASP + ABC + neighborhood search	Generalized TSP
Hernández-Pérez [31]	GRASP + variable neighborhood search	One-commodity TSP
Marinakis and Marinaki [32]	GRASP + ABC	Probabilistic TSP
Dai *et al*. [33]	AIS + quantum search	Generalized TSP
Dai *et al*. [34]	IA + bi-direction quantum search	Generalized TSP
Masutti and de Castro [35]	IA + ANN	Generalized TSP
Gao *et al*. [36]	IA + ACO	Generalized TSP
Guo *et al*. [37]	IA + greedy search	Generalized TSP
Gao *et al*. [38]	IA + chaotic search	Generalized TSP
Wei *et al*. [39]	ACO + chaotic search	Generalized TSP
Duan and Yu [40]	ACO + Memetic algorithm	Generalized TSP
Gunduz *et al*. [41]	ACO + ABC	Generalized TSP
Saenphon *et al*. [42]	ACO + opposite gradient search	Generalized TSP
Shuang *et al*. [43]	ACO + PSO	Generalized TSP
Elloumi *et al*. [44]	ACO + PSO	Generalized TSP
Mostafa *et al*. [45]	ACO + PSO + 3-opt	Generalized TSP
Saadatmand-Tarzjan *et al*. [46]	ANN (Hopfield) + ANN (self-organization map)	Generalized TSP
Muhammed and Tareq [47]	ANN (elastic net) + iterative improvement	Generalized TSP

demonstrate that pure algorithms are almost always inferior to hybrids.

To contribute more in this research area, this work investigates the effect of hybridizing IA with EDA as few investigations have been reported for such hybridization. A novel IA based on the clonal selection principle is utilized to perform the search, while EDA is used to construct a probability model for further sampling of new solutions. Two different probability updating models, i.e. the univariate marginal distribution algorithm (UMDA) and the population-based incremental learning (PBIL), are introduced based on the permutation representation of TSP. In addition, a refined local heuristic is also designed for those sampled new solutions to guarantee their feasibilities. The hybridization of IA with EDA, called, IA-EDA, is validated based on a number of TSP instances with size up to 100 000 cities. The simulation results indicate that IA-EDA is superior to the original IA and some other hybrid algorithms.

The remaining sections of the paper are organized as follows. Section 2 gives a brief description of IAs, while Section 3 describes the related background of EDA and its algorithmic framework. The proposed IA-EDA is described in detail in Section 4. In Section 5, extensive simulations results are presented. Finally, we give some general remarks and further works to conclude this paper.

2. Immune Algorithm

Artificial immune system [49] is one of the nature-inspired algorithms. It is a population-based problem-solving technique and mimics the mechanisms of the biological immune response, which depicts the procedures of responses when a biological immune system is exposed to an antigen. The most commonly used mechanisms of the biological immune system are clonal selection proliferation, negative selection, immune network, danger theory, and dendritic cell model [50,51]. Among them, the clonal selection algorithm is a special class of IA, and it is inspired by the clonal selection principle. Recently, clonal selection algorithm is very popular in the IA community and brings about a large number of applications, such as optimization, learning, clustering, and so on [52]. For solving optimization problems, clonal selection algorithm utilizes a collective learning process of a population of antibodies, and undergoes a cycle process of clonal proliferation, maturation, and antibody selection. Based on a fitness function, the clonal proliferation favors better antibodies to reproduce more

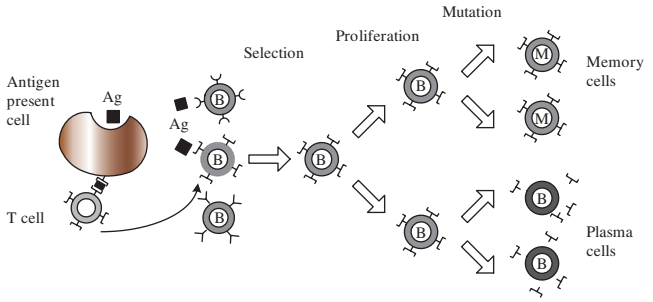


Fig. 1. Clonal selection principle

often than those are worse. During the period of maturation, descendants of antibodies are generated using randomized learning operators. Thereafter, fitter antibodies are selected to be reserved to enter into the next generation. The general clonal selection principle is illustrated in Fig. 1.

To date, several variants of the clonal selection algorithm have been developed for solving TSP. The first one is CLONALG [53], which is inspired by the two most important features of the affinity maturation in cells. One is that the proliferation of each antibody is proportional to its affinity, while the other is that the mutation suffered from an antibody is inversely proportional to its affinity. These two features make the clonal selection algorithm distinct from other evolutionary-like algorithms, such as GA. By introducing the immune memory strategy, Liu *et al*. [54] proposed an MCSA for TSP, and further developed it in [55]. After realizing the information exchange between different antibodies through receptor editing [56], idiotypic network mechanism [57], and the feedback interaction between B and T cells [58], the performance of the clonal selection algorithm was greatly improved. Moreover, attempts were made by incorporating other search algorithms such as ACO [36], chaotic search [38], greedy search [37], self-organizing map [35], and quantum search [33,34] into the conventional clonal selection algorithm to achieve high-performing hybrid algorithms.

In this paper, we adopt the simple receptor-editing-embedded clonal selection algorithm [56] as the immune algorithm (IA) to be combined with EDA. The reason for choosing this variant is twofold. First, compared with CLONALG, which can simultaneously solve continuous and discrete optimization problems, the receptor-editing-embedded clonal selection algorithm is a specialized IA for solving TSP where the receptor-editing operator is proposed specially for permutation-encoding-based problems. Second, no sophisticated (but generally time-consuming or needing complex data structure) mutation operators (such as Lin–Kernighan [59], edge assembly crossover [60], etc.) are utilized in the hybrid algorithm, suggesting that (i) the effect of hybridization of two algorithms can be clearly observed by comparing the hybrid one with the single-component algorithm, and (ii) potential improvement can be further achieved by incorporating such sophisticated mutation operators if adequate computational time burden can be afforded in the actual application.

3. Estimation of Distribution Algorithm

Estimation of distribution algorithm (EDA) is a new area of evolutionary computation, signaling a paradigm shift in genetic and evolutionary computation research [61]. Incorporating (automated) linkage learning techniques into a graphical probabilistic model, EDA exploits a feasible probabilistic model built around superior solutions found thus far while efficiently traversing the search space [62–65]. Thus, it has a theoretical foundation in probability theory and serves as a population-based search tool. An algorithmic framework of most EDAs can be described as follows:

Framework of EDA
Pop = Initialize population(); /*Initialization*/
while Stopping criteria are not satisfied **do**
Pop_{sel} = Select(Pop); /*Selection*/
$Prob$ = Estimate(Pop_{sel}); /*Estimation*/
Pop = Sample($Prob$); /*Sampling*/
end-while

At the beginning of the EDA search, a solution population Pop and a solution distribution model $Prob$ are initialized, and this is followed by a main search loop, consisting of three principal stages. The first stage is to select some best individuals from Pop, according to the fitness criteria. These individuals are used in the second stage in which the solution distribution model $Prob$ is updated or recreated. The third stage consists of sampling the updated solution distribution model to generate new solution offspring. These new solutions are evaluated and incorporated into the original population, replacing some or all of the old ones. This process is repeated until the termination criterion is met.

4. Hybridization of IA and EDA

It is observed from the description of IA and EDA that the two algorithms have different tendency to search the optimum solution. It is expected that the hybridization of the two distinct algorithms can take advantage of the characteristics of both and thus has the ability to overcome the inherent disadvantages of each component algorithm. The overall framework of the hybrid IA-EDA is shown in Fig. 2, where initialization, affinity evaluation, clone operator, hyper-mutation, receptor editing, probabilistic modeling, sampling, refinement operator, and apoptosis are iterated until a prespecified termination criterion is fulfilled.

4.1. Permutation Representation Various data structures can be used to code the tour for TSP, such as the *permutation* representation, the *matrix* representation, which is usually adopted in the ANN [66], and the *Tree* representation (involving Splay Tree [67], Two-Level Tree [68], and Segment Tree [69]) whose implementation is efficient but programming is more complex. Considering the simplicity of programming and the analogy with the gene representation in the immune system, the *Permutation* $\pi = (\pi_1, \pi_2, ..., \pi_i, ..., \pi_N)$ is utilized in this paper, where $\pi_i = k$ denotes that the city k is in the position i in the tour. Thus this permutation represents a feasible solution for TSP that the first city to be visited is the value of π_1 and the ith city to be visited is the value of π_i. The last city to be visited before going back to the city π_1 is the city π_N.

4.2. Modeling and Sampling of EDA This section describes the two different types of EDA including the univariate marginal distribution algorithm (UMDA) and the population-based incremental learning (PBIL). The original versions of the EDAs are in binary representation [61] which is not suitable for the permutation representation of TSP. Thus, the probabilistic modeling and sampling approaches are altered to manipulate the permutation representation for dealing with TSP.

UMDA makes use of univariate modeling for the learning of the probability distribution of the cities in each position of the permutation without consideration of the linkage dependencies between the cities [64]. The modeling is constructed based on an $N \times N$ probability matrix $P^t(\pi_i)$, which models the probability of the cities in TSP:

$$P^t = \begin{bmatrix} p^t(\pi_1 = c_1) & \dots & p^t(\pi_N = c_1) \\ p^t(\pi_1 = c_2) & \dots & p^t(\pi_N = c_2) \\ & \dots & \\ p^t(\pi_1 = c_N) & \dots & p^t(\pi_N = c_N), \end{bmatrix} \quad (2)$$

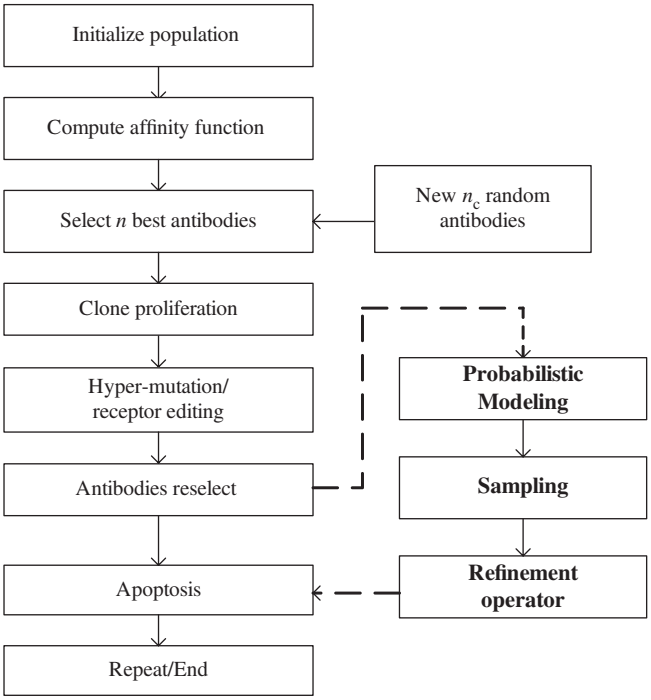


Fig. 2. Overall framework of the proposed IA-EDA

where P^t is the probability distribution of the cities at iteration t of the algorithm. $p^t(\pi_i = c_j)$ is the probability of city j to be located at the ith position of the permutation in an antibody. c_j denotes the city j ($c_j = j$), and N is the number of cities. The modeling considers the frequency of existence of the cities in each location of the antibody.

The probability of the cities in each position of the antibody is calculated according to the following equations:

$$p^t(\pi_i = c_j) = \frac{\sum_{k=1}^{m} \Delta_k(\pi_i^k = c_j) + 1/N}{m + m/N} \tag{3}$$

$$\Delta_k(\pi_i^k = c_j) = \begin{cases} 1 & if \ \ \pi_i^k = c_j \\ 0 & otherwise, \end{cases} \tag{4}$$

where m is the population size of antibodies, and $k = 1, ..., m$ denotes the order of the antibody in current iteration. It should be noted that the term $1/N$ is added to set the upper and lower bounds to the probability of each city. This is important, as the probability of 0.0 and 1.0 will make no progress in latter evolutions since a probability of 0.0 indicates that there will never be an iteration of this particular city in the position of the antibody, and this will generate infeasible solutions for TSP. Likewise, a probability of 1.0 suggests that there will always be the iteration the same city in the same position of the antibody, which will drastically decrease the diversity of the population, and thus will bring down the search performance of the algorithm.

On the other hand, PBIL [62] is another version of EDA that uses the same modeling approach as UMDA. The primary difference between PBIL and UMDA is in terms of its probability updating rule. To be specific, in PBIL, the probability of the cities in each position of the antibody is calculated by considering the probability of the cities in current and previous iterations of the evolution. The updating rule of PBIL can be shown as

$$pr^t(\pi_i = c_j) = \alpha p^t(\pi_i = c_j) + (1 - \alpha) p^{t-1}(\pi_i = c_j), \tag{5}$$

where $\alpha \in [0, 1]$ is the learning parameter of PBIL. $pr^t(\pi_i = c_j)$ is the eventual probability of the city j, which will be located at the ith position when sampling antibodies at the iteration t. $p^t(\pi_i = c_j)$ is obtained according to (3) and (4). α is set to 1.0 initially because there is no prior probability distribution from any previous iteration of evolution. In this situation, PBIL is similar to UMDA. To differentiate PBIL from UMDA, α would never be set to 1.0 over the course of the evolution process.

In addition, a sampling method based on roulette wheel selection is used for both UMDA and PBIL. New antibodies are generated by sampling the computed probability distribution, according to the following equation:

$$\pi_j = \begin{cases} c_1 & if \ \ r \le p^t(\pi_j = c_1) \\ c_2 & if \ \ p^t(\pi_j = c_1) \le r \le \sum_{i=1}^{2} p^t(\pi_j = c_i) \\ \dots & \\ c_N & if \ \ \sum_{i=1}^{N-1} p^t(\pi_j = c_i) \le r \le \sum_{i=1}^{N} p^t(\pi_j = c_i), \end{cases}$$

where π_j is a newly generated city at the jth position of an antibody, and r is a uniformly generated random number between 0 and 1.

4.3. Refinement Operator

It is important to point out that the new antibodies generated by EDA cannot guarantee feasibility. In other words, circumstances that some cities may not be visited at all or be visited more than once exist in the solutions sampled by EDA. Thus, a simple refinement local search operator is proposed not only to fix the feasibility of antibodies but also to improve the quality of these antibodies in a local search manner. The pseudo-code is described in Algorithm 1.

Algorithm 1–**Refinement operator**

For $i = 1 : Q$
input an antibody π^i sampled by EDA.
identify the location of the repeated cities in the antibody π^i, $L = \{L_1, ..., L_j, ..., L_R\}$.
identify the unvisited cities in π_i, $V = \{V_1, ..., V_k, ..., V_R\}$.
For $k = 1 : R$
calculate the probability p_{jk} for all unvisited cities V_k to be located at the position L_j according to the following equation:

$$p_{jk} = \frac{d'(\pi_{L_j - 1}, V_k)}{\sum_{l=1}^{R} (d'(\pi_{L_j - 1}, V_l))} \tag{6}$$

insert the unvisited city V_k to the L_j-th position of the antibody and discard that city from V.
End-For k
End-For i

In Algorithm 1, Q denotes the number of antibodies sampled by EDA, and R represents the number of repeated cities. The function $d'(X, Y) = 1/d(X, Y)$ denotes the reciprocal of the distance between city X and city Y. To explain Algorithm 1 in detail, for example, assume a TSP instance with $N = 7$ and an antibody $\pi = \{3, 2, 2, 4, 3, 1, 5\}$ generated by EDA. The antibody is clearly an infeasible solution for TSP. Using Algorithm 1, we got $R = 2$, $L = \{3, 5\}$ and $V = \{6, 7\}$. If the distance between cities 2 and 6 is smaller than that between 2 and 7 (i.e. $d(2, 6) < d(2, 7)$), then it is more possible to generate a refined solution $\{3, 2, 6, 4, 7, 1, 5\}$ than the other one $\{3, 2, 7, 4, 6, 1, 5\}$, and vice versa.

4.4. Structure of IA-EDA

IA-EDA works as follows.

Step 1. First, m antibodies are randomly generated to create an initial population, which can be represented as $(A_1, A_2, ..., A_m)$. To guarantee the feasibility of initial antibodies (i.e. solutions),

each antibody is generated based on a random permutation $(c_{i_1}, c_{i_2}, ..., c_{i_N})$.

Initialize the probability distribution model in (2) with all element in the matrix having the same value $1/N$, revealing that the probability model has no guiding information for the initial search.

Step 2. Calculate the affinity of all antibodies ($D(A_1)$, $D(A_2)$,..., $D(A_m)$)) according to (1).

Step 3. Select the n ($< m$) fittest antibodies based on their affinities.

Step 4. Clone all antibodies with a rate proportional to its affinity. The amount of clone generated for these antibodies is given by $(n - i) \times Q/n$, where Q denotes the clone size and i represents the rank of the antibody in the population.

Step 5. Mutate all antibodies under either the hypermutation operator or receptor editing. The hypermutation performs a random point mutation, while receptor editing carries out an inverse operator of the gene fragment [56].

Step 6. Update the population using mutated clones. The affinity of the offspring cloned from the same parent will be compared. Then the one with the highest affinity will be selected to replace its parent.

Step 7. Update the probability distribution model using UMDA described by (3) and (4) and PBIL by (5).

Step 8. Sample Q new antibodies based on (6).

Step 9. Perform the refinement operator introduced in Algorithm 1.

Step 10. Insert all newly sampled antibodies by EDA into the population and select the n best ones from it. The rest antibodies are removed from the population (i.e. apoptosis process).

Step 11. Replace the worst n_c (generally $n_c = \lceil 0.1 \times n \rceil$) with the new randomly generated antibodies to maintain the diversity of the population.

Step 12. Repeat Steps 4–11 until the maximum number of generations G_{max} to evolve is reached.

5. Experimental Results and Discussion

In this section, the performance of the proposed IA-EDA is investigated by applying the algorithm to solve TSP benchmark instances taken from the standard TSPLIB library [70]. There are three types of the selected instances involving EUC_2D, GEO, and ATT. They mean the Euclidean distance, geographical distance, and pseudo-Euclidean distance, respectively. A detailed description can be found in TSPLIB. Then we listed these instances in Table II and indicated their corresponding maximum numbers of generation used in our implementation. The tested benchmark instances are with sizes ranging from 51 to 100 000 cities. Moreover, each instance is run for 30 independent replications to make a statistical analysis.

The proposed IA-EDA was implemented in C++ language under Visual Studio 2010 and run on a PC with an Intel 1.70 GHz CPU. In addition to the maximum generation number, the other parameters of IA-EDA were fixed on the following values: the number of initial antibodies m = 100, the population size n = 50, the clonal size of proliferation Q = 50, the complementary intensity between hypermutation and receptor editing = 0.5 (i.e. two mutation operators have the same probability to be performed), and the learning parameter α of PBIL = 0.9. These parameters were experimentally obtained by testing the IA-EDA on the different instances.

Table II. Problem instances used in our simulation and the maximum number of generations for each instance

Instance	Size	Type	Optimum	G_{max}
eil51	51	EUC_2D	426	1000
st70	70	EUC_2D	675	1000
eil76	76	EUC_2D	538	1000
gr96	96	GEO	55209	1000
rd100	100	EUC_2D	7910	1000
eil101	101	EUC_2D	629	1000
lin105	105	EUC_2D	14379	1000
pr107	107	EUC_2D	44303	1000
pr124	124	EUC_2D	59030	1000
bier127	127	EUC_2D	118282	2000
pr136	136	EUC_2D	96772	2000
pr152	152	EUC_2D	73682	2000
rat195	195	EUC_2D	2323	2000
kroA200	200	EUC_2D	29368	5000
lin318	318	EUC_2D	42029	10000
pcb442	442	EUC_2D	50778	10000
att532	532	ATT	27686	20000
ja9847	9874	EUC_2D	491924	20000
mona-lisa	100000	EUC_2D	5757191	20000

5.1. Analysis of the Effects and Search Dynamics derived from EDA To investigate the effects of EDA on IA, two variants of IA-EDA and the original IA [56] were implemented on all tested instances. The IA-EDA using UMDA is denoted as IA-UMDA, while the other one using PBIL is named IA-PBIL. The characteristic difference between IA-UMDA and IA-PBIL is the updating mechanism of the probability distribution model. IA-UMDA utilizes only the current information of antibodies generated by IA to update the probability model, while IA-PBIL further makes use of the cumulative information of the probability model. For the sake of perspicuity, two assessment criteria of performance are used to analyze the experimental results. The optimum tour lengths that are listed in Table II are labeled D_{opt}. *PDM* and *PDB* which indicate the percentage deviation from the D_{opt} of the mean distance D_m and best distance D_b, respectively, were defined as follows:

$$PDM = \frac{D_m - D_{opt}}{D_{opt}} \times 100 \tag{7}$$

$$PDB = \frac{D_b - D_{opt}}{D_{opt}} \times 100 \tag{8}$$

Table III summarizes the experimental results for all tested TSP instances based on IA, IA-UMDA, and IA-PBIL, where the best solution obtained by the algorithm for each instance is highlighted in boldface, and 'T(s)' denotes the computational times in seconds. From Table III, it is apparent that IA-UMDA and IA-PBIL perform much better than the original IA on all the tested 19 instances, suggesting that the EDA technique (either UMDA or PBIL) is capable of enabling IA to find much better solutions. Furthermore, IA-PBIL can obtain the global optimal solution in 30 runs for 8 (out of 19) instances, and performs better than its two competitors for 16 instances except gr96, rat195, and kroA200. It indicates that, in most cases, it is better to utilize the cumulative information of the probability model to sample solutions in EDA. The average *PDM* and *PDB* for the 19 instances is 1.43 and 0.88, respectively. Thus, it can be concluded that IA-PBIL can produce better solutions than IA and IA-UMDA within reasonable computational times.

To further analyze the search dynamic of the algorithms and the obtained solutions, Fig. 3 depicts the comparative results by means of convergence graphs and box-and-whisker diagrams of

Table III. Experimental results of the original IA, and the two variants of IA-EDA, i.e. IA-UMDA and IA-PBIL

Problem	D_{opt}	IA			IA-UMDA			IA-PBIL		
		PDM	PDB	T(s)	PDM	PDB	T(s)	PDM	PDB	T(s)
eil51	426	2.59	0.94	4.0	0.13	0	7.1	0.12	**0**	7.5
st70	675	2.26	0.44	6.3	0.27	0	11.7	0.16	**0**	12.2
eil76	538	5.25	3.34	6.5	0.17	0	13.8	0.12	**0**	14.2
gr96	55209	4.95	2.73	6.7	0.81	**0.42**	17.6	0.76	0.54	18.4
rd100	7910	6.14	2.39	6.7	0.18	0	18.6	0.17	**0**	19.5
eil101	629	7.01	4.29	7.4	1.53	0.48	19.0	1.53	**0**	19.7
lin105	14379	4.98	1.45	7.3	0	0	19.3	0	**0**	19.9
pr107	44303	3.98	0.95	7.8	0.53	0.10	19.4	0.33	**0**	19.6
pr124	59030	6.50	2.82	7.9	0.80	0.10	19.7	0.59	**0.10**	20.1
bier127	118282	5.15	2.60	16.0	0.51	0.25	39.2	0.51	**0.04**	41.1
pr136	96772	7.33	5.07	16.9	1.25	0.20	41.4	0.95	**0.20**	43.5
pr152	73682	4.29	2.07	17.6	0.23	0	43.6	0.09	**0**	47.5
rat195	2323	14.1	11.2	14.8	1.45	**0.47**	48.9	1.41	0.77	51.2
kroA200	29368	7.90	6.32	38.2	1.28	**0.49**	177.3	1.31	0.67	184.1
lin318	42029	10.0	7.91	102.9	2.74	2.06	601.2	2.48	**1.55**	645.7
pcb442	50778	16.3	13.2	133.5	3.91	2.99	789.5	3.69	**2.71**	801.6
att532	27686	11.5	10.3	306.2	3.73	3.00	1576.1	3.56	**2.98**	1656.4
ja9847	491924	14.9	12.5	403.4	6.75	4.57	2045.2	4.77	**4.25**	2457.1
mona-lisa	5757191	21.5	19.6	501.5	9.71	6.53	3345.6	4.58	**2.86**	3865.3
Average:		8.24	5.80	84.8	1.89	1.14	466.0	1.43	0.88	523.4

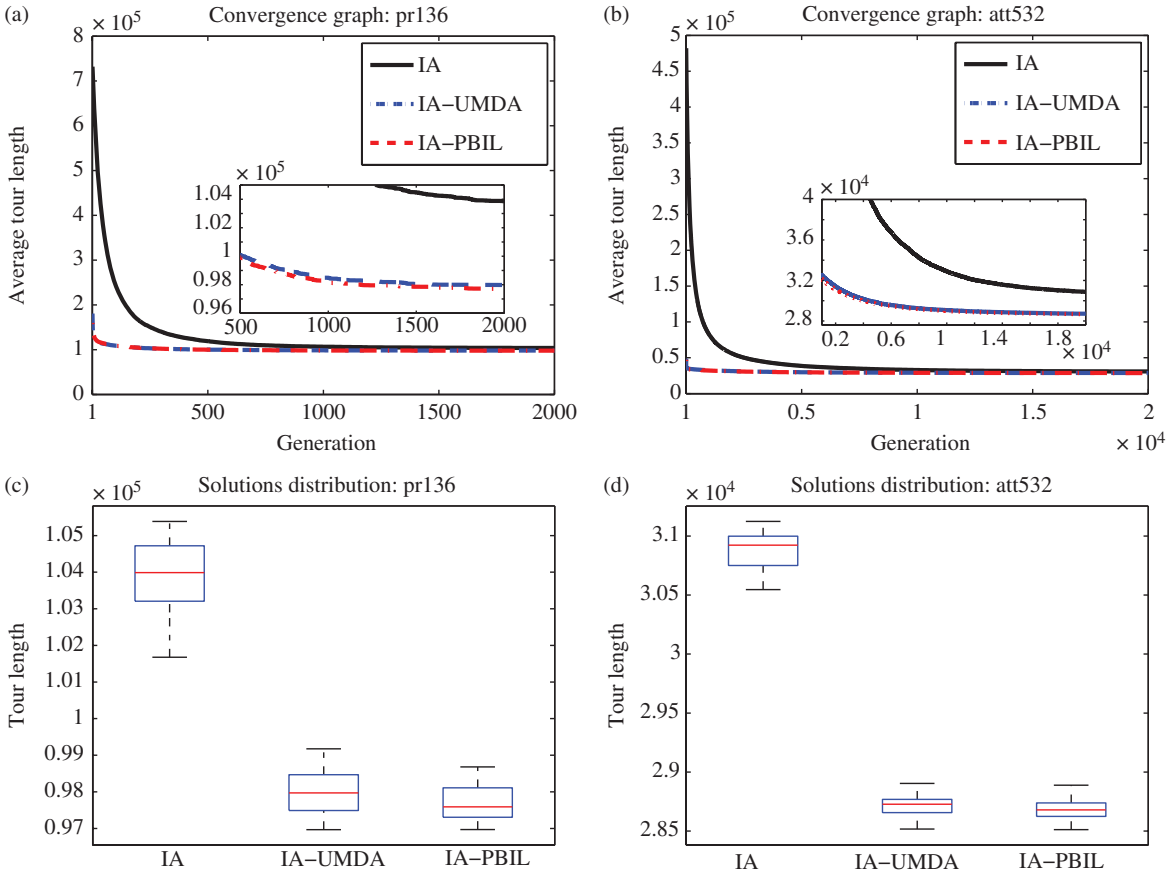


Fig. 3. Comparison of the average convergence performance and the distribution of the obtained solutions using box-and-whisker plots during IA, IA-UMDA, and IA-PBIL over 30 replication runs for instances pr136 and att532, respectively

solutions during IA, IA-UMDA, and IA-PBIL over 30 replication runs for instances st70, pr136, and att532. Similar results can be acquired for the other instances also. The algorithms' behaviors on the selected three instances are quite illuminating to further elaborate the effects of EDA on the search dynamics of the algorithm. It should be noted that the convergence graphs in Fig. 3 are plotted after the first generation, indicating that all solutions obtained by IA-UMDA or IA-PBIL have been manipulated after EDA sampling and refinement for one time. Thus, the tour length starts from a smaller value in IA-UMDA (or IA-PBIL) than in IA.

To be specific, IA-UMDA and IA-PBIL can produce significantly better solutions than IA after the first generation although all of them use randomly generated initial populations. In other words, EDA together with the heuristic refinement local search

Table IV. Results of the Wilcoxon's signed rank test at the level of significance $\alpha = 0.05$ and the average rankings of the algorithms obtained by the Friedman test on all tested TSP instances

	IA-PBIL vs IA				IA-PBIL vs IA-UMDA				IA-UMDA vs IA						
Problem	R^+	R^-	p-Value	Sig.	R^+	R^-	p-Value	Sig.	R^+	R^-	p-Value	Sig.	IA	IA-UMDA	IA-PBIL
eil51	465.0	0.0	0	YES	248.5	216.5	0.375	NO	465.0	0.0	0	YES	3	1.5333	1.4667
st70	465.0	0.0	0	YES	285.0	180.0	0.216	NO	465.0	0.0	0	YES	3	1.5667	1.4333
eil76	465.0	0.0	0	YES	244.5	220.5	0.755	NO	465.0	0.0	0	YES	3	1.5	1.5
gr96	465.0	0.0	2.0E-6	YES	262.5	172.5	0.323	NO	465.0	0.0	2.0E-6	YES	3	1.5833	1.4167
rd100	465.0	0.0	2.0E-6	YES	195.0	240.0	1	NO	465.0	0.0	2.0E-6	YES	3	1.45	1.55
eil101	465.0	0.0	0	YES	203.0	232.0	1	NO	465.0	0.0	0	YES	3	1.4167	1.5833
lin105	465.0	0.0	2.0E-6	YES	232.5	232.5	0.992	NO	465.0	0.0	2.0E-6	YES	3	1.5	1.5
pr107	465.0	0.0	2.0E-6	YES	359.5	75.5	0.002	YES	465.0	0.0	2.0E-6	YES	3	1.6833	1.3167
pr124	465.0	0.0	2.0E-6	YES	337.5	127.5	0.024	YES	465.0	0.0	2.0E-6	YES	3	1.7	1.3
bier127	465.0	0.0	2.0E-6	YES	216.5	248.5	1	NO	465.0	0.0	2.0E-6	YES	3	1.4	1.6
pr136	465.0	0.0	2.0E-6	YES	317.0	148.0	0.080	NO	465.0	0.0	2.0E-6	YES	3	1.5667	1.4333
pr152	465.0	0.0	2.0E-6	YES	361.5	73.5	0.000	YES	465.0	0.0	2.0E-6	YES	3	1.75	1.25
rat195	465.0	0.0	1.0E-6	YES	248.0	187.0	0.476	NO	465.0	0.0	2.0E-6	YES	3	1.55	1.45
kroA200	465.0	0.0	2.0E-6	YES	243.0	222.0	0.821	NO	465.0	0.0	1.0E-6	YES	3	1.5667	1.4333
lin318	465.0	0.0	2.0E-6	YES	326.5	138.5	0.051	NO	465.0	0.0	2.0E-6	YES	3	1.6667	1.3333
pcb442	465.0	0.0	2.0E-6	YES	299.5	135.5	0.074	NO	465.0	0.0	2.0E-6	YES	3	1.5833	1.4167
att532	465.0	0.0	2.0E-6	YES	316.0	149.0	0.084	NO	465.0	0.0	2.0E-6	YES	3	1.6	1.4
ja9847	465.0	0.0	2.0E-6	YES	360.0	75.0	0.002	YES	465.0	0.0	2.0E-6	YES	3	1.7	1.3
mona-lisa	465.0	0.0	2.0E-6	YES	380.5	54.5	0.000	YES	465.0	0.0	2.0E-6	YES	3	1.8	1.2

operator can greatly improve the quality of the initial solutions. The reason seems to be twofold: (i) since there is no information exchange procedure in IA, EDA can realize the information exchange through the probability model during different antibodies, thus being able to sample promising solutions for IA, and (ii) the refinement local search operator utilizes the problem-dependent knowledge (at least the distance matrix of the TSP instance), which can also facilitate the search to some extent. Generally, IA-UMDA and IA-PBIL show faster convergence than IA. Although the convergence graphs of IA-UMDA and IA-PBIL exhibit almost the same search dynamics, IA-PBIL can produce better solutions than IA-UMDA.

Further considerations deal with the significant differences in the behavior of IA, IA-UMDA, and IA-PBIL. Table IV summarizes the results of the Wilcoxon signed rank test [71] and the average rankings of the algorithms obtained by the Friedman test [72,73] on all tested TSP instances. In Table IV, R^+ denotes the sum of ranks for the problems in which the base algorithm (i.e. the one before 'vs') outperformed the competitive one (i.e. the latter), and R^- the sum of ranks for the opposite. The associated asymptotic p-values are also computed to identify whether a statistical hypothesis test is significant or not. It is worth pointing out that, from the statistical point of view, the Wilcoxon signed rank test is more sensitive and safer than the t-test, because it does not assume normal distributions, and, meanwhile, the outliers have less effect on the Wilcoxon test than on the t-test [71]. Moreover, the Friedman test applying the *post hoc* method of Iman–Davenport [73] is another nonparametric statistical test that can rank the algorithms for each problem separately; the best performing algorithm should have rank 1, the second best rank 2, etc. The values in the last three columns record the average ranking of the three algorithms for each TSP instance. As Table IV shows, IA-PBIL and IA-UMDA show significant improvement over IA at a level significance $\alpha = 0.05$ for all 19 TSP instances. IA-PBIL performs significantly better than IA-UMDA on pr107, pr124, pr152, ja9847, and mona-lisa. Friedman test shows that IA performs the worst among the three algorithms, and IA-PBIL acquires a total average rank of 1.41, which is slightly smaller than of the value (1.58) obtained by IA-UMDA, indicating that IA-PBIL performs statistically better than IA-UMDA.

To make a clear illustration of the solutions obtained by IA-EDA (either IA-UMDA or IA-PBIL), Fig. 4 shows the best routes found by IA-EDAs (three obtained by IA-UMDA and the others obtained by IA-PBIL) and their corresponding total tour length.

5.2. Analysis of the Influence of the Refinement Operator The proposed refinement operator plays an important role in the search process of the algorithm, not only enabling the antibodies sampled by EDA to be feasible but also effectively utilizing the distance knowledge of TSP by a local 'greedy search' method shown in Eq. (6). To give more insights into the influence of the refinement operator, we also employed two other competitors: one is a variant of IA-EDA wherein the refinement operator is modified by using a random insertion method, and called IA-RiEDA for short; while the other is a simpler hybrid algorithm (called IA-R), which only makes use of the refinement operator in IA.

To be more specific, the random insertion method of the refinement operator used in IA-RiEDA can be described as follows. In all infeasible antibodies sampled by EDA, each unvisited city will be inserted into the location of the repeated city randomly. To realize this, we use a uniformly generated number to replace the probability generated by (6). Consequently, two kinds of IA-RiEDA are conducted, namely IA-RiUMDA and IA-RiPBIL, by using UMDA and PBIL, respectively. Through the comparison between IA-EDA and IA-RiEDA, we expect to find out the effects of the 'greedy search' manner derived from (6) over that of the newly established random insertion refinement operator on the search performance.

On the other hand, IA-R is used for finding out the influence of the refinement operator on IA, and further for inferring which component (the refinement operator or EDA) is more effective when combined with IA. As the refinement operator in Algorithm 1 is initially proposed for refining infeasible antibodies sampled by EDA, it is revised in IA-R since all solutions generated by IA are feasible. IA-R only needs to perform the refinement operator as a local search operator rather than a refinement technique. Thus, in the experiment, we modify Algorithm 1 as in the following. First, randomly select a position i ($i = 1, 2, ..., N$) in the antibody

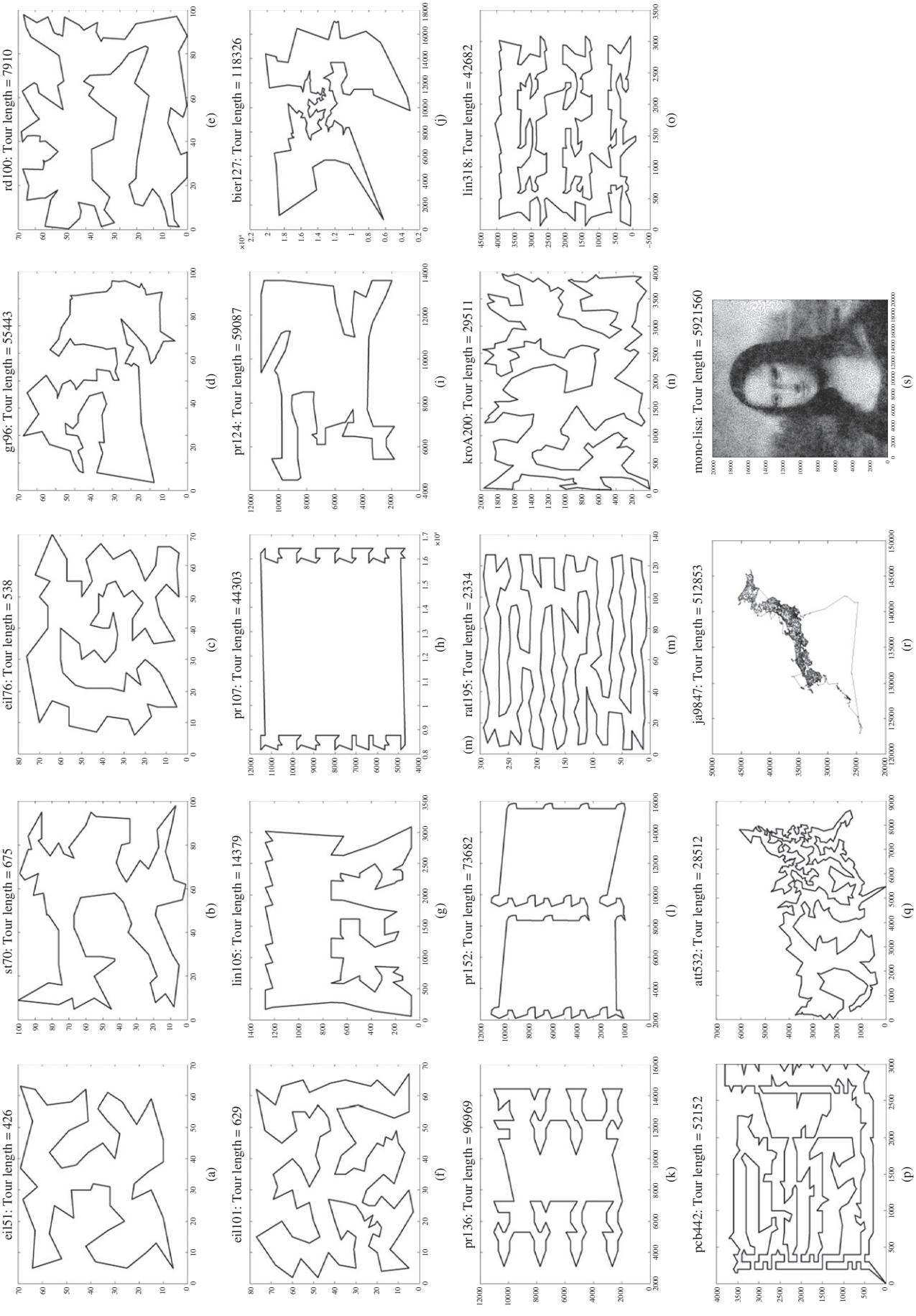


Fig. 4. Best routes found by IA-EDA for the tested 19 TSP instances

Table V. Experimental results of IA, IA-R, and the two variants of IA-RiEDA, i.e. IA-RiUMDA and IA-RiPBIL

	IA			IA-R			IA-RiUMDA			IA-RiPBIL		
Problem	PDM	PDB	T(s)	PDM	PDB	T(s)	PDM	PDB	T(s)	PDM	PDB	T(s)
eil51	2.59	0.94	4.0	0.66	0	4.7	0.37	0	7.0	0.31	0	7.1
st70	2.26	0.44	6.3	1.02	0	9.7	0.98	0.30	11.5	0.50	0	11.6
eil76	5.25	3.34	6.5	2.03	0.74	11.9	0.69	0	13.2	0.53	0	13.6
gr96	4.95	2.73	6.7	2.13	0.54	12.1	1.29	0.42	17.3	0.88	0.55	17.5
rd100	6.14	2.39	6.7	2.75	0.11	12.6	0.87	0.04	18.1	0.75	0.05	18.3
eil101	7.01	4.29	7.4	3.31	2.38	12.6	3.38	2.38	18.2	3.45	2.07	18.5
lin105	4.98	1.45	7.3	2.23	0.90	12.6	0.09	0	18.9	0.02	0	19.2
pr107	3.98	0.95	7.8	1.51	0.51	12.3	1.39	0.57	19.0	0.93	0.41	19.4
pr124	6.50	2.82	7.9	2.65	0.89	13.2	1.00	0.22	19.1	0.74	0.10	19.6
bier127	5.15	2.60	16.0	2.91	1.87	26.4	1.13	0.23	38.4	1.02	0.54	39.1
pr136	7.33	5.07	16.9	3.69	2.42	28.9	3.07	1.72	40.5	2.62	1.06	42.7
pr152	4.29	2.07	17.6	1.47	0.18	30.6	0.83	0.01	42.5	0.23	0	45.8
rat195	14.1	11.2	14.8	8.40	5.90	22.6	2.69	1.76	48.0	2.61	1.38	50.1
kroA200	7.90	6.32	38.2	4.15	1.88	85.9	2.75	1.87	172.8	2.30	1.50	180.2
lin318	10.0	7.91	102.9	6.12	4.69	184.3	4.70	3.06	598.7	4.04	2.79	637.3
pcb442	16.3	13.2	133.5	9.02	6.22	269.2	6.95	5.74	777.8	6.29	4.55	796.0
att532	11.5	10.3	306.2	6.55	4.68	559.9	6.02	4.32	1502.5	5.81	4.21	1603.6
ja9847	14.9	12.5	403.4	10.09	9.26	694.5	8.36	5.52	1964.2	7.56	6.91	2238.2
mona-lisa	21.5	19.6	501.5	18.45	16.77	876.6	12.54	9.47	3300.4	11.76	9.02	3512.7
Average:	8.24	5.80	84.8	4.69	3.16	151.6	3.11	1.98	454.1	2.76	1.85	489.0

generated by IA, and then select another position j according to the following probability:

$$p_j = \frac{d'(\pi_i, \pi_j)}{\sum_{k=1}^{N} (d'(\pi_i, \pi_k))}, \tag{9}$$

where $d'(\pi_i, \pi_j) = 1/d(\pi_i, \pi_j), (j = 1, 2, ..., N \ and \ j \neq i)$ is the reciprocal of the distance between city π_i and city π_j. According to (9), the gene position that is nearer to the preselected gene position has a higher probability to be selected, thus also exhibiting the same 'greedy search' manner as in Algorithm 1. For an antibody $\pi = \{\pi_1, \pi_2, ..., \pi_N\}$ generated by IA, the refinement operator used in IA-R can be illustrated as follows:

$$\pi_1 \to \pi_2 ... \to \pi_{\mathbf{i}} \to \pi_{i+1} ... \to \pi_{\mathbf{j}} \to \pi_{j+1} ... \to \pi_N$$
$$\Downarrow$$
$$\pi_1 \to \pi_2 ... \to \pi_{\mathbf{i}} \to \pi_{\mathbf{j}} \to \pi_{i+1} ... \to \pi_{j+1} ... \to \pi_N$$

Table V summarizes the comparative results between the newly constructed hybrid algorithms (IA-R, IA-RiUMDA, IA-RiPBIL) and the original IA for all tested TSP instances. Intuitively, IA-R, IR-RiUMDA, and IA-RiPBIL perform better than IA in terms of the average and best solution qualities. In detail, their average *PDB* for 19 instances is 3.16, 1.98, and 1.85, respectively, which is much smaller than that generated by IA (5.80). The results show that either EDA with random insertion method or the refinement operator with (9) is beneficial for the search in IA. Furthermore, the two variants of IA-RiEDA can generate better solutions than IA-R, suggesting that the effects derived from EDA are more significant than that derived from the refinement operator. Nevertheless, IA-RiEDA is still inferior to IA-EDA by comparing their experimental results, thus again confirming the effectiveness of the proposed algorithm. On the other hand, we plot the convergence graph and solution distribution for all compared algorithms, namely IA, IA-R, IA-RiUMDA, IA-RiPBIL, IA-UMDA, and IA-PBIL, on instances st70 and lin318 in Fig. 5. Similar results can also be illustrated on other TSP instances. It can be seen from Fig. 5 that IA-R converges much faster than IA, indicating that the 'greedy search' mechanism employed by the refinement operator can enable IA to find better solutions quickly. All in all, it can be concluded that IA-EDA (either IA-UMDA or IA-PBIL) can perform better than its variants in terms of convergence speed and solution quality.

5.3. Performance Comparison with Other Hybrid Meta-heuristics Finally, an intensive comparison with the existing six hybrid meta-heuristics in the literature is carried out to further validate the performance of the proposed IA-EDAs. Table VI summarizes the computational results of the two variants of IA-EDA and its six competitors in terms of the average tour length (Avg.), the standard deviation of the obtained tour lengths, and the *PDM* defined in (8), which is used to reflect the percentage relative error. As can be seen from Table VI, the proposed IA-EDA performs better than its competitors on st70, rd100, and bier127, and can generate very competitive solutions for the rest of the instances, suggesting that the proposed hybrid meta-heuristic search algorithm by combining IA with EDA is a promising alternative method for solving TSP. In addition, it is worth pointing out that the complexity of construction of IA-EDAs is significantly less than that of the two algorithms in Refs. [23,45] since there are more than two partial components in the latter hybrid algorithms, although two algorithms in [23,45] can produce slightly better solutions than IA-EDAs.

6. Conclusions

This study proposed combining EDA and IA to create a new hybrid that can efficiently solve TSPs. In this method, EDA is used to realize the information exchange during different solutions generated by IA through the probabilistic model. EDA enables IA to quickly converge on promising search areas. To refine the solutions sampled by EDA, a heuristic local search operator was also proposed to repair the infeasible solutions and further facilitate the search by making use of the problem-dependent knowledge of TSP. The effects of two kinds of EDAs, i.e. the univariate marginal distribution algorithm and the population-based incremental learning, were investigated by taking into consideration the average tour length, the best tour length, convergence performance, and the distribution of solutions on 19 different TSP instances with size up to 100 000 cities. A nonparametric statistical test based on the Wilcoxon signed rank

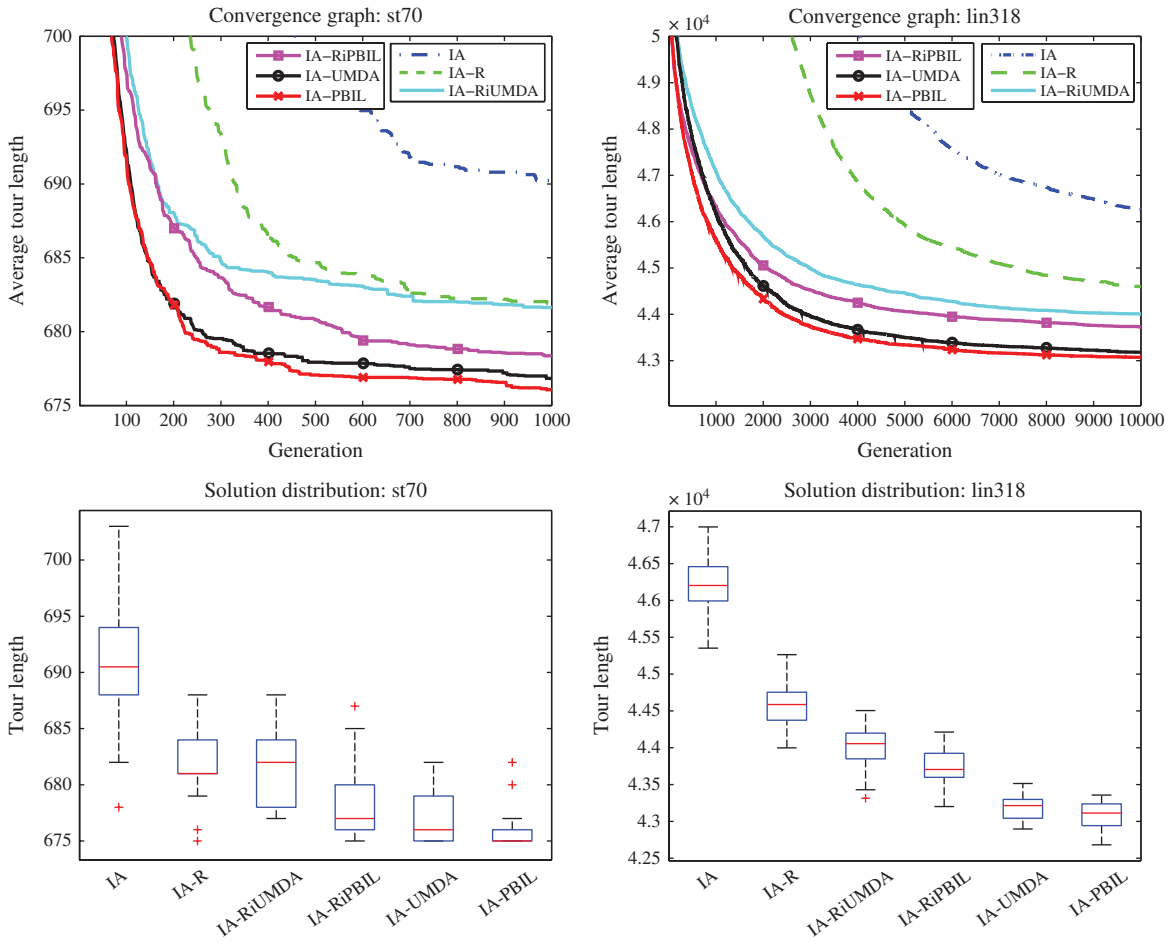


Fig. 5. Comparison of the average convergence performance and the distribution of the obtained solutions using box-and-whisker plots during IA, IA-R, IA-RiUMDA, IA-RiPBIL, IA-UMDA, and IA-PBIL over 30 replication runs for instances st70 and lin318, respectively

Table VI. Computational results of the two variants of the proposed IA-EDA and its competitor algorithms in the literature. D_{opt} is the best known solution; Avg. is the average tour length; SD is the standard deviation; *PDM*, defined in (8), shows the percentage relative error

Method	Problem D_{opt}	eil51 426	st70 675	eil76 538	rd100 7910	eil101 629	lin105 14379	bier127 118282	kroA200 29368	lin318 42029
GA+ACO (2004) [10]	Avg.	430.68	–	555.70	–	–	–	–	–	–
	SD	–	–	–	–	–	–	–	–	–
	PDM	1.10	–	3.29	–	–	–	–	–	–
IA+ANN (2009) [35]	Avg.	437.47	–	556.33	8199.77	648.63	14400.17	120886.33	30190.27	43696.87
	SD	4.2	–	5.30	80.77	3.85	44.03	1158.79	273.38	410.06
	PDM	2.69	–	3.41	3.66	3.12	0.15	2.20	2.80	3.97
GA+SA+ACO+PSO (2011) [23]	Avg.	427.27	–	540.20	7987.57	635.23	14406.37	119421.83	29738.73	**43002.90**
	SD	0.45	–	2.94	62.06	3.59	37.28	580.83	356.07	307.51
	PDM	0.30	–	0.41	0.98	0.99	0.19	0.96	1.26	2.32
CGA+ACO (2012) [14]	Avg.	–	–	542.00	–	–	–	–	29946.00	–
	SD	–	–	–	–	–	–	–	–	–
	PDM	–	–	0.74	–	–	–	–	1.97	–
ACO+ABC (2015) [41]	Avg.	443.39	700.58	557.98	–	683.39	–	–	–	–
	SD	5.25	7.51	4.10	–	6.56	–	–	–	–
	PDM	4.08	3.79	3.71	–	8.65	–	–	–	–
ACO+PSO+3-opt (2015) [45]	Avg.	**426.45**	678.20	**538.30**	–	**632.70**	14379.15	–	**29646.05**	–
	SD	0.61	1.47	0.47	–	2.12	0.48	–	114.71	–
	PDM	0.11	0.47	0.06	–	0.59	0	–	0.95	–
IA + UMDA (proposed method)	Avg.	426.57	676.83	538.90	7924.47	638.60	14379.00	**118886.63**	29744.30	43182.13
	SD	0.68	2.35	1.47	21.07	3.33	0	176.59	113.68	154.07
	PDM	0.13	0.27	0.17	0.18	1.53	0	0.51	1.28	2.74
IA + PBIL (proposed method)	Avg.	426.53	**676.07**	538.67	**7923.90**	638.63	**14379.00**	118888.73	29753.07	43072.27
	SD	0.68	1.91	1.15	22.97	4.05	0	264.95	125.22	192.72
	PDM	0.12	0.16	0.12	0.17	1.53	0	0.51	1.31	2.48

test and the Friedman test consistently demonstrated the effects of EDA on IA. Intensive comparative results of other six hybrid metaheuristics reported recently in the literature verified the superiority of IA-EDA and showed that IA-EDA could produce better or competitive solutions compared to other hybrid algorithms within reasonable computational times.

Finally, it is important to remark that the proposed algorithm assessed in this work did not make use of 'strong' operators, in the sense that no efficient route improvement heuristics available to date, such as Lin–Kerninghan [59], edge assembly crossover [60], or quantum interference crossover [33], were incorporated to perform the search. In spite of that, the proposed algorithm was capable of finding solutions that are on average less than 1% worse than the best known results for all tested instances, thus showing the strength of the hybrid strategy.

In future, we plan to apply the proposed IA-EDA to solve more optimization problems such as the vehicle routing problem and the sequential ordering problem.

Acknowledgment

This research was partially supported by the National Natural Science Foundation of China (Grant Nos. 61203325, 11572084, 11472061, and 61472284), the Shanghai Rising-Star Program (No. 14QA1400100), the Natural Science Foundation Programs of Shanghai (No. 13ZR1443100), and JSPS KAKENHI Grant No. 15K00332 (Japan).

References

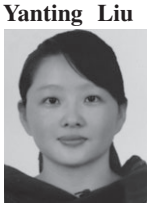
(1) Gutin G, Punnen AP. *The Traveling Salesman Problem and Its Variations*. Springer: Springer, NM2002.
(2) Laporte G. The traveling salesman problem: an overview of exact and approximate algorithms. *European Journal of Operational Research* 1992; **59**(**2**):231–247.
(3) Ergun O, Orlin JB. A dynamic programming methodology in very large scale neighborhood search applied to the traveling salesman problem. *Discrete Optimization* 2006; **3 1**:78–85.
(4) Padberg M, Rinaldi G. Branch-and-cut approach to a variant of the traveling salesman problem. *Journal of Guidance, Control, and Dynamics* 1988; **11**(**5**):436–440.
(5) Wolpert DH, Macready WG. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation* 1997; **1**(**1**):67–82.
(6) Deng W, Chen R, He B, Liu Y, Yin L, Guo J. A novel two-stage hybrid swarm intelligence optimization algorithm and application. *Soft Computing* 2012; **16**(**10**):1707–1722.
(7) Wang RL, Zhou XF, Okazaki K. Ant colony optimization with genetic operation and its application to traveling salesman problem. *IEICE Transactions on Fundamentals of Electronics Communications and Computer Sciences* 2009; **E92A**(**5**):1368–1372.
(8) Mavrovouniotis M, Yang SX. A memetic ant colony optimization algorithm for the dynamic travelling salesman problem. *Soft Computing* 2011; **15**(**7**):1405–1425.
(9) Baraglia R, Hidalgo JI, Perego R. A hybrid heuristic for the traveling salesman problem. *IEEE Transactions Evolutionary Computation* 2001; **5**(**6**):613–622.
(10) Tsai CF, Tsai CW, Tseng CC. A new hybrid heuristic approach for solving large traveling salesman problem. *Information Sciences* 2004; **166**(**1**):67–81.
(11) Nguyen HD, Yoshihara I, Yamamori K, Yasunaga M. Implementation of an effective hybrid ga for large-scale traveling salesman problems. *IEEE Transactions on System, Man and Cybernetics, Part B: Cybernetics* 2007; **37**(**1**):92–99.
(12) Xing LN, Chen YW, Yang KW, Hou F, Shen XS, Cai HP. A hybrid approach combining an improved genetic algorithm and optimization strategies for the asymmetric traveling salesman problem. *Engineering Applications of Artificial Intelligence* 2008; **21**(**8**):1370–1380.
(13) Lin W, Delgado-Frias JG, Gause DC, Vassiliadis S. Hybrid Newton-Raphson genetic algorithm for the traveling salesman problem. *Cybernetics and System* 1995; **26**(**4**):387–412.
(14) Dong G, Guo WW, Tickle K. Solving the traveling salesman problem using cooperative genetic ant systems. *Expert systems with applications* 2012; **39**(**5**):5006–5011.
(15) Abbattista F, Abbattista N, Caponetti L. An evolutionary and cooperative agents model for optimization. IEEE International Conference on Evolutionary Computation, 1995; 668–671.
(16) Lope HS, Coelho LS. Particle swarn optimization with fast local search for the blind traveling salesman problem. Fifth International Conference on Hybrid Intelligent Systems, 2005; 245–250.
(17) Chen H, Flann NS. Parallel simulated annealing and genetic algorithms: a space of hybrid methods. In *Parallel Problem Solving from Nature*. Davidor Y, Schwefel H-P, Manner R (eds). Springer: Berlin and Heidelberg; 1994; 428–438.
(18) Creput JC, Koukam A. A memetic neural network for the euclidean traveling salesman problem. *Neurocomputing* 2009; **72**(**4**): 1250–1264.
(19) Jin HD, Leung KS, Wong ML, Xu ZB. An efficient self-organizing map designed by genetic algorithms for the traveling salesman problem. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics* 2003; **33**(**6**):877–888.
(20) Glover F, Kelly JP, Laguna M. Genetic algorithms and tabu search: hybrids for optimization. *Computers & Operations Research* 1995; **22**(**1**):111–134.
(21) Shim VA, Tan KC, Chia JY, Chong JK. Evolutionary algorithms for solving multi-objective travelling salesman problem. *Flexible Services and Manufacturing Journal* 2011; **23**(**2**):207–241.
(22) Marinakis Y, Migdalas A, Pardalos PM. A hybrid genetical grasp algorithm using lagrangean relaxation for the traveling salesman problem. *Journal of Combinatorial Optimization* 2005; **10**(**4**):311–326.
(23) Chen SM, Chien CY. Solving the traveling salesman problem based on the genetic simulated annealing ant colony system with particle swarm optimization techniques. *Expert Systems with Applications* 2011; **38**(**12**):14439–14450.
(24) Martin OC, Otto SW. Combining simulated annealing with local search heuristics. *Annals of Operations Research* 1996; **63**(**1**):57–75.
(25) Malek M, Guruswamy M, Pandya M, Owens H. Serial and parallel simulated annealing and tabu search algorithms for the traveling salesman problem. *Annals of Operations Research* 1989; **21**(**1**):59–84.
(26) Geng XT, Chen ZH, Yang W, Shi DQ, Zhao K. Solving the traveling salesman problem based on an adaptive simulated annealing algorithm with greedy search. *Applied Soft Computing* 2011; **11**(**4**):3680–3689.
(27) Pepper JW, Golden BL, Wasil E. Solving the traveling salesman problem with annealing-based heuristics: a computational study. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans* 2002; **32**(**1**):72–77.
(28) Shim VA, Tan KC, Cheong CY. A hybrid estimation of distribution algorithm with decomposition for solving the multiobjective multiple traveling salesman problem. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews* 2012; **42**(**5**):682–691.
(29) Marinakis Y, Marinaki M. A hybrid multi-swarm particle swarm optimization algorithm for the probabilistic traveling salesman problem. *Computers & Operations Research* 2010; **37**(**3**):432–442.
(30) Marinakis Y, Marinaki M, Dounias G. Honey bees mating optimization algorithm for the euclidean traveling salesman problem. *Information Sciences* 2011; **181**(**20**):4684–4698.
(31) Hernandez-Perez H, Rodriguez-Martin I, Salazar-Gonzalez JJ. A hybrid grasp/vnd heuristic for the one-commodity pickup-and-delivery traveling salesman problem. *Computers & Operations Research* 2009; **36**(**5**):1639–1645.
(32) Marinakis Y, Marinaki M. A hybrid honey bees mating optimization algorithm for the probabilistic traveling salesman problem. IEEE Congress on Evolutionary Computation, 2009; 1762–1769.
(33) Dai HW, Yang Y, Li CH, Shi J, Gao SC, Tang Z. Quantum interference crossover-based clonal selection algorithm and its application to traveling salesman problem. *IEICE Transactions on Information & Systems* 2009; **E92-D**(**1**):78–85.
(34) Dai H, Yang Y, Li H, Li C. Bi-direction quantum crossover-based clonal selection algorithm and its applications. *Expert Systems with Applications* 2014; **41**(**16**):7248–7258.
(35) Masutti TA, de Castro LN. A self-organizing neural network using ideas from the immune system to solve the traveling salesman problem. *Information Sciences* 2009; **179**(**10**):1454–1468.

(36) Gao SC, Wang W, Dai HW, Li FJ, Tang Z. Improved clonal selection algorithm combined with ant colony optimization. *IEICE Transactions on Information & Systems* 2008; **E91-D**(**6**):1813–1823.
(37) Pan G, Li K. Ouyang A, Li K, Hybrid immune algorithm based on greedy algorithm and delete-cross operator for solving tsp. *Soft Computing* 2014; 1–12.
(38) Gao SC, Tang Z, Dai HW, Zhang JC. An improved clonal selection algorithm and its application to traveling salesman problems. *IEICE Transactions Fundamentals of Electronics Communications and Computer Sciences* 2007; **E90-A**:2930–2938.
(39) Wei Z, Ge FZ, Lu Y, Li LX, Yang YX. Chaotic ant swarm for the traveling salesman problem *Nonlinear Dynamics* 2011; **65**(**3**):271–281.
(40) Duan H, Yu X. Hybrid ant colony optimization using memetic algorithm for traveling salesman problem. IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning, IEEE, 2007; 92–95.
(41) Gunduz M, Kiran MS, Ozceylan E. A hierarchic approach based on swarm intelligence to solve the traveling salesman problem. *TURKISH Journal of Electrical Engineering and Computer Sciences* 2015; **23**(**1**):103–117.
(42) Saenphon T, Phimoltares S, Lursinsap C. Combining new fast opposite gradient search with ant colony optimization for solving travelling salesman problem. *Engineering Applications of Artificial Intelligence* 2014; **35**:324–334.
(43) Shuang B, Chen J, Li Z. Study on hybrid ps-aco algorithm. *Applied Intelligence* 2011; **34**(**1**):64–73.
(44) Elloumi W, El Abed H, Abraham A, Alimi AM. A comparative study of the improvement of performance using a pso modified by aco applied to tsp. *Applied Soft Computing* 2014; **25**:234–241.
(45) Mahi M, Baykan OK, Kodaz H. A new hybrid method based on particle swarm optimization, ant colony optimization and 3-opt algorithms for traveling salesman problem. *Applied Soft Computing* 2015; **30**:484–490.
(46) Saadatmand-Tarzjan M, Khademi M, Akbarzadeh-T MR, Moghaddam HA. A novel constructive-optimizer neural network for the traveling salesman problem. *IEEE Transactions on Systems, Man and Cybernetics Part B: Cybernetics* 2007; **37**(**4**):754–770.
(47) Al-Mulhem M, Al-Maghrabi T. Efficient convex-elastic net algorithm to solve the euclidean traveling salesman problem. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics* 1998; **28**(**4**):618–620.
(48) Talbi EG. A taxonomy of hybrid metaheuristics *Journal of Heuristics* 2002; **8**(**5**):541–564.
(49) Dasgupta D, Yu S, Nino F. Recent advances in artificial immune systems: Models and applications. *Applied Soft Computing* 2011; **11**:1574–1587.
(50) Timmis J. Artificial immune systems: Today and tomorrow. *Natural Computing* 2007; **6**(**1**):1–18.
(51) Wang S, Gao SC, Todo Y, Tang Z. A multi-learning immune algorithm for numerical optimization. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences* 2015; **98**(**1**):362–377.
(52) Haktanirlar Ulutas B, Kulturel-Konak S. A review of clonal selection algorithm and its applications. *Artificial Intelligence Review* 2011; **36**(**2**):117–138.
(53) De Castro LN, Von Zuben FJ. Learning and optimization using the clonal selection principle. *IEEE Transactions on Evolutionary Computation* 2002; **6**(**3**):239–251.
(54) Liu RC, Jiao LC, Du HF. Clonal strategy algorithm based on the immune memory. *Journal of Computer Science and Technology* 2005; **20**(**5**):728–734.
(55) Liu R, Jiao L, Li Y, Liu J. An immune memory clonal algorithm for numerical and combinatorial optimization. *Frontiers of Computer Science in China* 2010; **4**(**4**):536–559.
(56) Gao SC, Dai H, Yang G, Tang Z. A novel clonal selection algorithm and its application to traveling salesman problem. *IEICE Transactions on Fundamentals of Electronics Communications and Computer Science* 2007; **E90-A**:2318–2325.
(57) Gao SC, Dai H, Zhang J, Tang Z. An expanded lateral interactive clonal selection algorithm and its application. *IEICE Transactions on Fundamentals of Electronics Communications and Computer Science* 2008; **E91-A**:2223–2231.
(58) Gao SC, Wang RL, Ishii M, Tang Z. An artificial immune system with feedback mechanisms for effective handling of population size. *IEICE Transactions on Fundamentals of Electronics Communications and Computer Science* 2010; **E93-A**:532–541.
(59) Karapetyan D, Gutin G. Lin-kernighan heuristic adaptations for the generalized traveling salesman problem *European Journal of Operational Research* 2011; **208**(**3**):221–232.
(60) Tsai HK, Yang JM, Tsai YF, Kao CY. An evolutionary algorithm for large traveling salesman problems. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics* 2004; **34**(**4**): 1718–1729.
(61) Pelikan M, Hauschild MW, Lobo FG. Estimation of distribution algorithms. In *Springer Handbook of Computational Intelligence*. Kacprzyk J, Pedrycz W(eds). Springer: Berlin and Heidelberg; 2015; 899–928.
(62) Hauschild M, Pelikan M. An introduction and survey of estimation of distribution algorithms. *Swarm and Evolutionary Computation* 2011; **1**(**3**):111–128.
(63) Chen CH, Chen YP. Quality analysis of discretization methods for estimation of distribution algorithms. *IEICE Transactions on Information & Systems* 2014; **97**(**5**):1312–1323.
(64) Larranaga P, Lozano JA. *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*. Kluwer Academic Press: 2001.
(65) Shim VA, Tan KC, Cheong CY. A hybrid estimation of distribution algorithm with decomposition for solving the multiobjective multiple traveling salesman problem. *IEEE Transactions on System, Man, and Cybernetics, Part C: Applications and Reviews* 2012; **42**(**5SI**): 682–691.
(66) Hopfield JJ, Tank DW. Neural computation of decision in optimization problems. *Biological Cybernetics*, 1985; **52**:141–152.
(67) Sleator DD, Tarjan RE. Self-adjusting binary search trees. *Journal of Association for Computing Machinery* 1985; **32**:652–686.
(68) Chrobak M, Szymacha T, Krawczyk A. A data structure useful for finding hamiltonian cycles. *Theoretical Computer Science*, 1990; **71**:419–424.
(69) Fredman ML, Johnson DS, McGeoch LA, Ostheimer G. Data structure for traveling salesmen. *Journal of Algorithms* 1995; **18**:432–479.
(70) Reinelt G. Tsplib–a traveling salesman problem library. *ORSA Journal on Computing* 1991; **3**:376–384.
(71) Derrac J, Garcia S, Molina D, Herrera F. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation* 2011; **1**:3–18.
(72) Garcia S, Fernandez A, Luengo J, Herrera F. Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power. *Information Sciences* 2010; **180**(**10**): 2044–2064.
(73) Alcala-Fdez J, Sanchez L, Garcia S, Del Jesus MJ, Ventura S, Garrell JM, Otero J, Romero C, Bacardit J, Rivas VM. Keel: a software tool to assess evolutionary algorithms for data mining problems. *Soft Computing* 2009; **13**(**3**):307–318.

Zhe Xu (Non-member) received the M.S. degree from the University of Toyama (UT), Toyama, Japan, in 2011. He is currently working toward the Ph.D. degree in the Faculty of Engineering, UT. His research interests include artificial immune algorithms, evolutionary algorithms, and combinatorial optimizations.

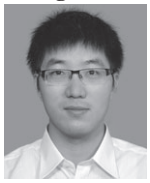
Yirui Wang (Non-member) received the B.S. degree from Donghua University (DU), Shanghai, China, in 2014. He is currently working toward the M.S. degree at the College of Information Sciences and Technology, DU. His current research interests include computational intelligence, swarm intelligent algorithms, and combinatorial optimizations.

Sheng Li (Non-member) received the B.S. degree from Nanjing University of Science and Technology, Nanjing, China, in 2006, and the M.S. and D.E. degrees from the University of Toyama, Toyama, Japan in 2009 and 2012, respectively. From 2012 to 2014, he worked as a Senior System Engineer with Lossev Technology Co., Ltd., Japan. Now he is a Lecturer with Taizhou University, China. His research interests include intelligence computing, neural networks, swarm intelligent algorithms, and combinational optimization problems.



Yanting Liu (Non-member) received the M.S. degree from the University of Toyama (UT), Toyama, Japan, in 2015. She is currently working toward the Ph.D. degree at the Faculty of Engineering, UT. Her research interests include evolutionary algorithms, complex networks, and optimization.

Yuki Todo (Non-member) received the B.S. degree from Zhejiang University, Zhejiang, China, the M.S. degree from Beijing University of Posts and Telecommunications, Beijing, China, and the D.E. degree from Kanazawa University, Kanazawa, Japan, in 1983, 1986, and 2005, respectively. From 1987 to 1989, she was an Assistant Professor with the Institute of Microelectronics, Shanghai Jiaotong University, Shanghai, China. From 1989 to 1990, she was a research student at Nagoya University, Nagoya, Japan. From 1990 to 2000, she was a Senior Engineer with Sanwa Newtech Inc., Miyazaki, Japan. From 2000 to 2011, she worked with Tateyama Systems Institute, Toyama, Japan. In 2012, she joined Kanazawa University, where she is now an Associate Professor with the School of Electrical and Computer Engineering. Her current research interests include multiple-valued logic, neural networks, and optimization.



Shangce Gao (Member) received the B.S. degree from Southeast University, Nanjing, China, in 2005, and the M.S. and Ph.D. degrees in intellectual information systems and innovative life science from the University of Toyama (UT), Toyama, Japan in 2008 and 2011, respectively. He is currently an Associate Professor with the Faculty of Engineering, UT. From 2011 to 2012, he was an Associate Research Fellow with the Key Laboratory of Embedded System and Service Computing, Ministry of Education, Tongji University, Shanghai, China. From 2012 to 2014, he was an Associate Professor with the College of Information Sciences and Technology, Donghua University, Shanghai, China. His research interests include computational intelligence, nature-inspired technologies, swarm intelligence, and neural networks for optimization and real-world applications. Dr Gao was a recipient of the Shanghai Rising-Star Scientist award, the Chen-Guang Scholar of Shanghai award, the Outstanding Academic Performance Award of IEICE, and the Outstanding Self-financed Students Abroad Award of the Chinese Government, and the Outstanding Academic Achievement Award of IPSJ.