Soft Comput
DOI 10.1007/s00500-015-1886-z

METHODOLOGIES AND APPLICATION

Imperial competitive algorithm with policy learning for the traveling salesman problem

Meng-Hui Chen[1] · **Shih-Hsin Chen**[2] · **Pei-Chann Chang**[1]

Abstract The traveling salesman problem (TSP) is one of the most studied combinatorial optimization problems. In this paper, we present the new idea of combining the imperial competitive algorithm with a policy-learning function for solving the TSP problems. All offspring of each country are defined as representing feasible solutions for the TSP. All countries can grow increasingly strong by learning the effective policies of strong countries. Weak countries will generate increasingly excellent offspring by learning the policies of strong countries while retaining the characteristics of their own country. Imitating these policies will enable the weak countries to produce improved offspring; the solutions generated will, therefore, acquire a favorable scheme while maintaining diversity. Finally, experimental results for TSP instances from the TSP library have shown that our proposed algorithm can determine the salesman's tour with more effective performance levels than other known methods.

Keywords Traveling salesman problem · Imperial competitive algorithm · Combinatorial optimization problems · Artificial chromosomes · Genetic algorithm

Communicated by V. Loia.

✉ Pei-Chann Chang
iepchang@saturn.yzu.edu.tw

[1] Department of Information Management, Innovation Center for Big Data and Digital Convergence, Yuan Ze University, Chung-Li, Taiwan

[2] Department of Information Management, Cheng Shiu University, Kaohsiung, Taiwan

1 Introduction

The traveling salesman problem (TSP) has been extensively studied and researched and is a well-known NP-hard problem in COPs. The problem is stated as follows: given *n* cities and the geographical distance between pairs of cities, the objective is to find the shortest closed tour in which all cities are visited exactly once. GAs were introduced by Holland (1973). These algorithms are adaptive search techniques based on the mechanisms of natural selection and the survival of the fittest concept of biological evolution. They have been used successfully in a variety of different problems, including the traveling salesman problem. The benefit of evolutionary computing is not only its simplicity but also its ability to obtain global optima. Many research findings have indicated that a well-adapted genetic local search algorithm can acquire a near-optimal solution better than those found by local search algorithms (Goldberg 1989; Pham and Karaboga 2000). Therefore, numerous results on evolutionary optimization have been published in recent years (Larranaga et al. 1999; Mohammadian et al. 2002). Using genetic algorithms to solve the traveling salesman problem (TSP) is one of the popular approaches (Larranaga et al. 1999).

In recent years, solving the TSP using evolutionary algorithms and specially GAs has attracted a lot of attention. Many studies have been performed and researchers attempt to further improve the effectiveness and efficiency of the GAs by contributing to different parts of solution processes. Some of researchers propose different forms of GA operators (Yan et al. 2005) and others attempt to combine GA with other possible approaches like ACO (Lee 2004), PSO, etc. Some researchers implement a new evolutionary idea or combine some previous algorithms and idea to create a new method (Bonyadi et al. 2007; Chang et al. 2010, 2013). In 2010,

Published online: 01 October 2015

Chang et al. (2010) first proposed an approach, namely artificial chromosomes for genetic algorithm (ACGA), which has been successful at injecting artificial chromosomes into the evolutionary process of the genetic algorithm and speeding up the convergence. In addition, Wang et al. (2011) proposed two types of artificial chromosome operators. In their method, the algorithm can evolve after a number of iterations of injections, improving the individual's ability to search for a suitable combination of chromosomes. The mechanism provides a more expansive searching space while evolving.

As pointed out by some researches, an effective algorithm should learn the positional and interactional information between the variables and to sample new solutions from the models (Pan and Ruiz 2012). In this paper, we propose an improved imperial competitive algorithm (ICA) which is embedded in a block-based artificial chromosome generation approach (Chang and Chen 2014) to further improve the solution quality with more competitive artificial chromosomes. The proposed approach is named as BBICA standing for Block Based Imperial Competitive Algorithm. According to the concept of ICA, different kingdoms will have different approaches of generating their offspring by policies. Therefore, blocks considered as policies from the probability matrices will be generated for each kingdom. BBICA adopt the concept of ICA to retain the diversity of the population and to avoid the convergence to local optima.

The primary idea of this paper is that using the ICA with different group recombination mechanisms to generate competitive artificial chromosomes increases the search space. The proposed approach is also enhanced by the local search using a hybrid of two mutation methods. The ICA simulates a multi-agent algorithm. Each agent is similar to a kingdom consisting of countries, with the strongest country in each agent being called the imperialist and the others being called colonies. Each country competes with the imperialist of the same kingdom by evolving. A country moves in a search space to find suitable solutions with a high fitness level, endeavoring to become the new imperialist. Allowing different kingdoms to evolve can increase the search space thus generating more competitive artificial chromosomes to speed up the convergence.

Finally, to demonstrate the proposed approach is effective. A series of experiments were conducted and the instances were taken from the TSP library. The experimental results show the comparison of BBICA with three types of mutant strategies, i.e., BBEA (Huang et al. 2012), RABNET-TSP (Pasti and Castro 2006), and SME (Somhom et al. 1997) and one state-of-the-art approach HMMA (Yong 2015).

2 Literature review

2.1 Traveling salesman problem

The TSP is a well-known NP-hard problem with many real-world applications such as job-shop scheduling, large-scale integrated routing (Gutin and Punnen 2002), vehicle routing (Laporte 1992), and drilling problems (Onwubolu and Clerc 2004). The TSP has been extensively used as a basic problem for comparison in improving different optimization techniques such as genetic algorithms (Affenzeller and Wanger 2003), simulated annealing (Budinich 1996), tabu search (Liu et al. 2003), local search (Bianchi et al. 2005), ant colony (Chu et al. 2004), and neural networks (Leung et al. 2004).

The TSP is a generalized version of the Hamilton cycle. The TSP consists of a complete graph with each edge having a nonnegative cost. The vertices refer to cities; edges refer to the path between cities, and edge costs refer to the distance between cities. The TSP can be initiated as follows: given n cities and the geographical distance between all pairs of these cities, the task is to find the shortest closed tour in which each city is visited exactly once. More formally, given n cities, the TSP requires searching for a permutation using cost matrix $D = \{d_{ij}\}$, where d_{ij} denotes the cost (known to the salesmen) of traveling from city i to city j, which minimizes the path length defined as follows:

$$f(\pi, D) = \sum_{i=0}^{n-1} d_{\pi_i, \pi_{(i+1)}} + d_{\pi_n, \pi_0}, \tag{1}$$

where π_i denotes the city at the ith location in the tour. Assuming that a city is marked by its position (x_i, y_j) in the plant and that cost matrix D contains a Euclidean distance between the ith and jth cities, the function is defined as follows:

$$d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \tag{2}$$

The objective of a salesman is to move from city to city, visiting each city only once and returning back to the starting city. This progression is called the salesman's tour. In the mathematical formulation, there is a group of distinct cities $\{C_1, C_2 \ldots C_N\}$, and there is a given for each pair of cities $\{C_i, C_j\}$ and a distance $d\{C_i, C_j\}$. The objective is to find an order π of cities, such that the total time for the salesman's tour is minimized. The objective function is defined as follows:

$$\sum_{i=1}^{N-1} d(C_{n(i)}, C_{n(i+1)}) + d(C_{n(N)}, C_{n(1)}) \tag{3}$$

This quality is known as the tour length. Two branches of this problem exist: asymmetric and asymmetric. A symmetric problem is one in which the distance between two cities is identical, written as $d\{C_i, C_j\} = d\{C_j, C_i\}$; an asymmetric problem is depicted as $d\{C_i, C_j\} \neq d\{C_j, C_i\}$. In this paper, we consider the symmetric TSP in which the distance from one city to another city is the same as the distance in the opposite direction. The mathematical formulation for the TSP is listed as follows:

$$\min \sum d_{ij} x_{ij} \tag{4}$$

$$\sum_{j=1}^{n} x_{ij} = 1 \quad i = 1, 2, \ldots n \tag{5}$$

$$\sum_{i=1}^{n} x_{ij} = 1 \quad j = 1, 2, \ldots n \tag{6}$$

$$\sum_{i,j \in S} x_{ij} \leq |S| - 1 \quad 2 \leq |S| \leq n - 2, \quad S \subset \{1, 2, \ldots n\} \tag{7}$$

$$x_{ij} \in \{0, 1\} \quad i, j = 1, 2, \ldots n \; i \neq j, \tag{8}$$

where d_{ij} is the distance between city i and city j, decision variable x_{ij} is the route the salesman passes through (including the route from city i to city j), and $x_{ij} = 0$ means that the route is not chosen by the salesman. Function (4) is the minimum total distance; Functions (5) and (6) only provide an assurance that the salesman visits each city once. Function (7) requires that no loop in any city subset should be formed by the salesman; $|S|$ represents the number of elements included in the set S.

In 1997, Johnson and McGeoch (1997) proposed an effective survey for solving the TSP. These methods can be approximately categorized as local search and global search approaches. Generally, the local search approach produces a result that is efficient and has a fast convergence (e.g., 2-opt and 3-opt) because of the selection, which consists of reconnecting cities on the basis of geometric neighborhood information and the edges from other individuals in the population. However, the salesman using this approach may be stuck at local minima because this approach does not effectively address the diversity of feasible solutions. Recently, two new approaches are proposed by Ouaarab et al. (2015), and Yong (2015). Ouaarab et al. (2015) used a simplified random-key encoding scheme to pass from a continuous space (real numbers) to a combinatorial space. They also considered the displacement of a solution in both spaces using Lévy flights. The hybrid Max–Min ant system (MMA) proposed by Yong (2015) integrated with four vertices and three lines inequality is then introduced to solve the TSP problem. In addition, Cheng et al. (2012) also applied Ant colony algorithm for solving the multi-objective TSP problem. In this research, each sub-colony independently optimizes its corresponding sub-problem and shares their evolutionary information.

2.2 Imperial competitive algorithm (ICA)

ICA is the mathematical model and the computer simulation of human social evolution, whereas GAs are based on the biological evolution of species. The concept of ICA is shown in Fig. 1.

As shown in Fig. 1, ICA is simulated using the competitive situation of three kingdoms, and the kingdom has its own countries. The strongest country in the kingdom is the empire, and other countries are the colonies. The strong kingdom will plunder the weak one. Finally, ICA is often designed to be end when there is only one kingdom. During the end of the cycle, all countries will improve their power, i.e., increasing their fitness, by learning from the empire. If there is any country stronger than its empire, the country will replace its empire and become the new empire.

The ICA is an evolutionary algorithm in the evolutionary computation field based on the imperialistic competition (Atashpaz-Gargari and Lucas 2007). Recently, Nozarian and Jahan (2012) proposed an improved memetic algorithm for solving metaheuristic problems. The method uses ICA as local search, and, similar to other evolutionary algorithms, the ICA starts with initial populations and each solution is defined as a country. In the BBICA, "Kingdom" denotes an artificial chromosome generation approach. Countries with different characteristics are generated by different kingdom.

3 Methodology

The ICA in this research is started by generating initial countries. Then we set group of 25 % countries of all as a kingdom, and the country with the highest fitness of each kingdom is considered as the empire. In order to implement policy-learning, we develop a probability matrix to collect the good information from the elite countries from each kingdom. Each kingdom will generate new countries by the policy, and then after the imperialistic competition, the resource for generating new countries of each kingdom would be reset by the law of jungle.

In this research, we designed four strategies to generate artificial chromosomes. In order to be advisable to assign the number of chromosomes to each empire, imperialistic competition of ICA is applied to control the four strategies to

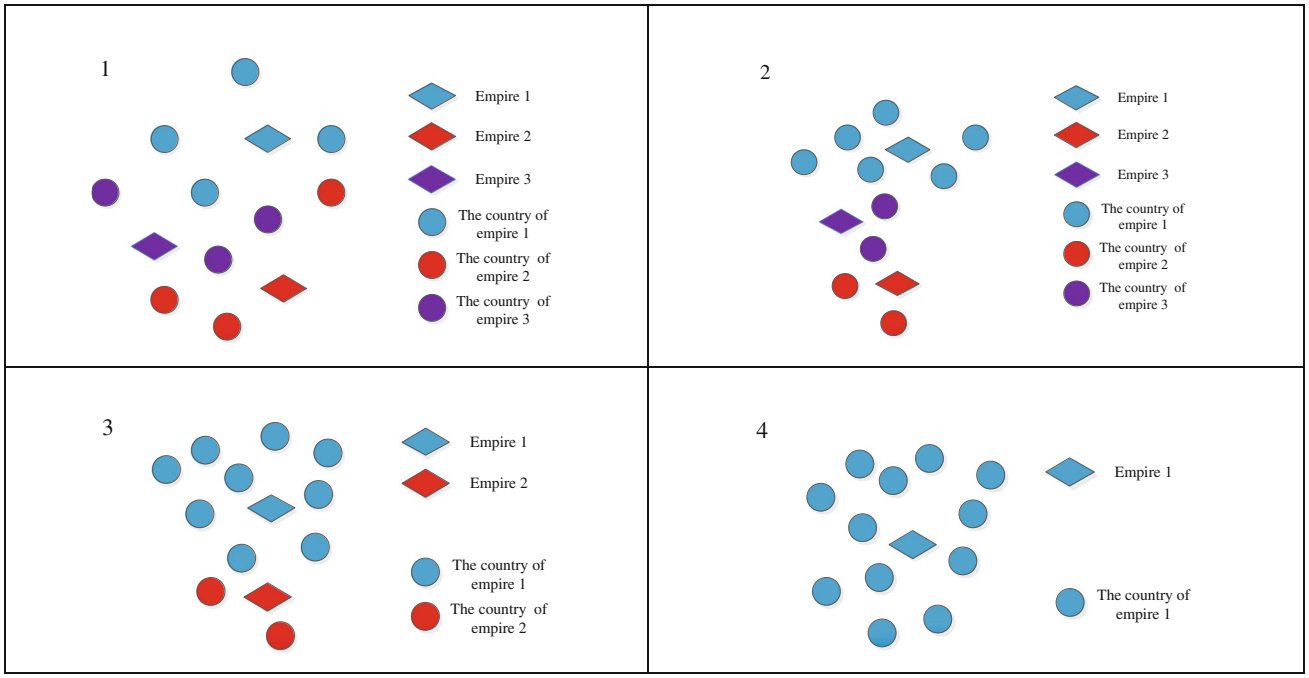

Fig. 1 The concept of ICA

generate the number of chromosomes. In addition, we collect the information from the solutions with high quality to construct the dependency probability matrix. Then we also apply the block which is generated from dependency probability matrix to implement the policy for each kingdom to generate new artificial chromosomes.

The BBICA aims to develop an effective metaheuristic algorithm containing two phases. The first phase adopts the mining-blocks approach to discover effective blocks. Meanwhile, these blocks are recombined during the course of the competition; the blocks of high quality are kept. The second phase develops the approach for composing the artificial chromosomes. In this study, we designed four approaches for composing artificial chromosomes. Every artificial chromosome composition represents a country; the number of generations given to a country is determined by its strength. In BBICA, we have four approaches to generate artificial chromosomes. These four approaches are represented as four kingdoms. Benefiting from the shortest distance or probability, we designed the approach in this study to combine the competitive blocks, which were randomly selected. However, not all of the blocks are used. The remaining cities are combined into new artificial chromosomes that are more competitive. Subsequently, we evaluate the reassembled solutions and select the solutions with favorable performance levels. In BBICA, we allow the solutions to mutate and consider the reassembled solutions, selecting the solutions with high performance levels for updating the probability matrix. The process is repeated until the defined iterations are satisfactory.

A flowchart describing the process of the BBICA is shown in Fig. 2. The initial solutions are generated randomly. The fitness of each solution is then calculated. We update the probability matrix by selecting the solution that is more competitive, based on their performance levels. During the evolutional process, the time for injecting the artificial chromosomes is determined by the convergence angle and iteration number. Subsequently, we mine the blocks according to the probability matrix. After mining the blocks, we inject the artificial chromosomes, which are combined with the blocks. The combined blocks continue to be adopted for the assembling of the new artificial chromosomes; the blocks in each artificial chromosome are randomly generated. To retain the diversity of the population, we run a mutation operator, according to the rank of each country, and we assign them a suitable mutation function. Finally, the new population is generated for the next evolution, until the stopping criteria are achieved.

In the remainder of Sect. 3, we introduce the details of probability matrix, the block mechanism, artificial chromosome generation, and local search strategies.

3.1 Dependency probability matrix

In this section, we demonstrate how to construct the dependency probability matrix. Each element in the dependency probability matrix is defined as in Eq. (9). To build up the dependency matrix, a set of m better chromosomes (C^1,C^2, …, C^m) are selected from the current generation t. Y_{ij}^k is a

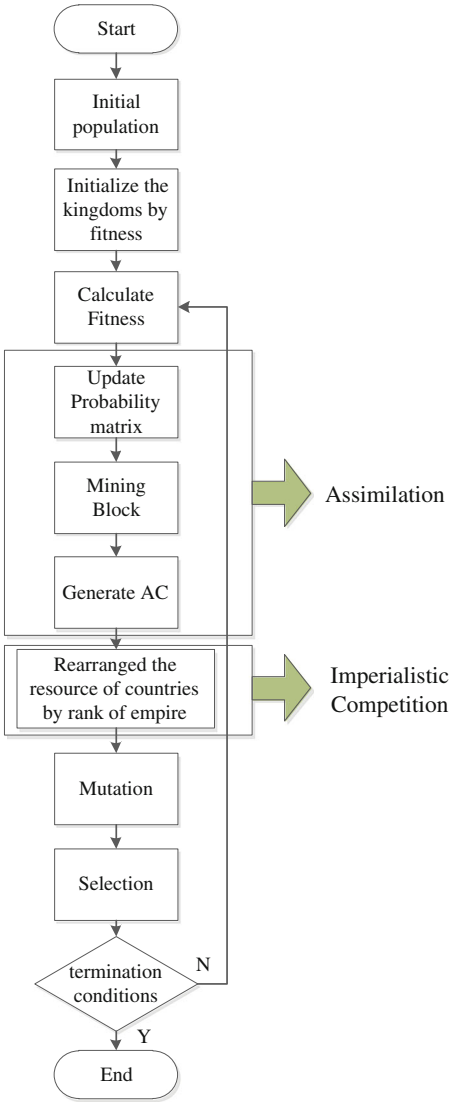

Fig. 2 The flowchart of BBICA

binary variable and can be treated like a gene within chromosome C^k.

$$Y_{ij}^k = \begin{cases} 1 & \text{if job } i \text{ is next to job } j \\ 0 & \text{otherwise} \end{cases}, \quad i = 1 \ldots n;\ j = 1 \ldots n;\ k = 1 \ldots m \tag{9}$$

Then $\theta_{ij}(t)$ in Eq. (10) which represents the number of times that city i is next to city j at the current generation t is found from summing up the statistic information from all m chromosomes to the Y_{ij}^k. The total number of generations is G.

$$\theta_{ij}(t) = \sum_{k=1}^{m} Y_{ij}^k, \quad i = 1 \ldots n;\ j = 1 \ldots n;\ t = 1 \ldots G;\ k = 1 \ldots m \tag{10}$$

We transform the dependency matrix into dependency probability matrix, and each element of the dependency probability matrix is defined as follows:

$$P_{ij}(t) = \frac{\theta_{ij}(t)}{m \times t} \quad i = 1 \ldots n;\ j = 1 \ldots n;\ t = 1 \ldots G \tag{11}$$

3.2 Block mechanism

Block is represented a policy for each kingdom referred to generate the artificial chromosomes. Thus the quality of blocks is important to affect the quality of artificial chromosomes. In this research, we stored the structure of solutions to build up a dependency probability matrix to generate blocks. A block mining procedure applies the probability matrix for extracting the blocks from the set of high-fit chromosomes. It is a process of block learning that is used to discover the hidden knowledge within the dependent variables. A block consists of a series of genes linked to each other continually. This section is organized as follows: Sect. 3.2.1 explains the block mining procedure. Section 3.2.2 develops a block competition mechanism.

3.2.1 Block mining procedure

A static block with size K can be generated according to the following procedures, which function according to two approaches in the BBICA: in the beginning of the first 10 % iterations, the starting city in a block is selected randomly and the top three cities with short distance to the starting city are selected as the candidate city of the second city in the block. The second city in the block is selected with the highest probability according to the dominance matrix. To demonstrate how to mine blocks in the first approach, a simple example including ten cities is shown in Fig. 3. City 3 is selected randomly, and Cities 9, 8, and 5 are the cities with short distance to City 3. Then City 9 has the highest probability according to the dominance matrix. So we can mine a block which is {3, 9}. By the way, there are several blocks constructed and stored in block archive.

In the remainder of iterations, the quality of solutions becomes better and better. Thus, the second strategy of min-

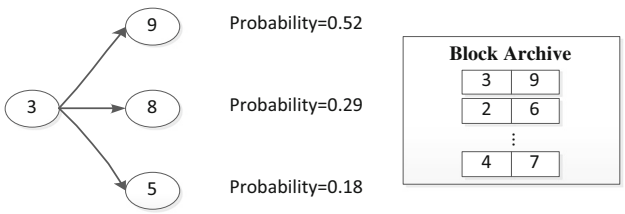

Fig. 3 The set of blocks mined from the distance

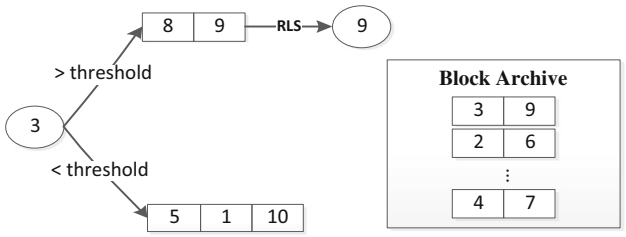

Fig. 4 The set of blocks mined from the probability matrix

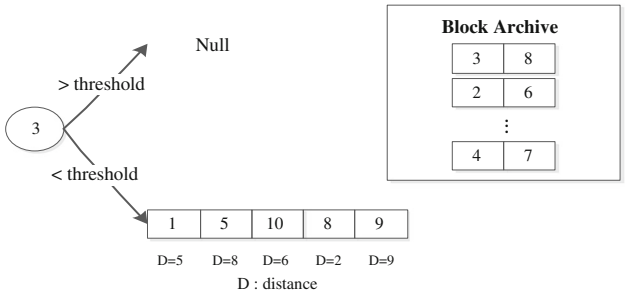

Fig. 5 The set of blocks mined from the probability matrix

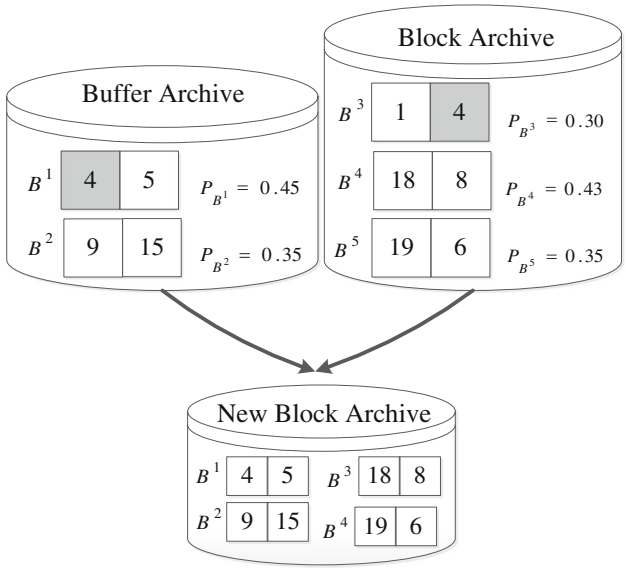

Fig. 6 The block competition mechanism

ing blocks is to set a minimum probability to be a threshold which is used to select a city to be the second city in a block. We also make a simple example including ten cities and the size of the blocks is 2 to demonstrate how the second strategy works. As shown in Fig. 4, the starting city in a block is also selected randomly. According to probability matrix, the probability of City 8 and City 9 is higher than the threshold we set, and then we use roulette wheel selection (RLS) to select the second city in the block.

In addition, if there is no city with probability higher than a certain threshold, the second city in the block is selected. As shown in Fig. 5, the distance of City 3 next to City 8 is the shortest. So, we can build a block {3, 8}.

On the basis of the two approaches, numerous blocks will be generated. To maintain high quality of blocks, we designed a mechanism to remove the inefficient blocks. To ensure that the quality of the blocks yield improved results, we use a block selection strategy. The strategy is to calculate the probability of each block according to the probability matrix and then sort the calculated values. In this study, only the remaining 20 % of the blocks are selected according to values of rank.

3.2.2 Block competition mechanism

As the countries evolve, there are two situations that generate blocks. In one situation, the numbers of the blocks may become increasingly high in value. In the other situation, the blocks may have the same cities. The proposed BBICA has a mechanism to prevent the blocks from exhibiting high performance levels in enhancing the evolution of countries and from having the same cities. The block competition mechanism is shown in Fig. 6.

The block archive stores the current blocks, and the buffer archive stores the new ones. When there are blocks with the same cities in the two archives, we evaluate the total numbers of each city probability in the blocks. If the total number is higher for a block, it indicates that the block has a higher performance level for enhancing evolutionary solutions. For example, B^1 and B^3 have the same "City 4," and B^1 has higher probabilities. Thus, B^1 would be saved, and B^3 would be abandoned.

3.3 Artificial chromosome generation

In Sect. 2.2, we mentioned that a "kingdom" denotes an artificial chromosome generation approach. We adopted imperialistic competition of ICA to control the four kingdoms to generate the number of chromosomes. Each kingdom generates artificial chromosomes as new countries by referring the blocks. An artificial chromosome is generated according to certain procedures, for which four approaches are used in the BBICA. As shown in Fig. 7, the following proposed procedure is the rule of the first kingdom:

1. A city is picked randomly as the head.
2. A city is selected as the continuous city from the residual cities using roulette-wheel selection. When the selected city is contained in a block, the block, instead of the selected city, is selected and placed in this chromosome. The last city of the block serves as the new head.
3. Step 2 is repeated until the number of residual cities is zero.

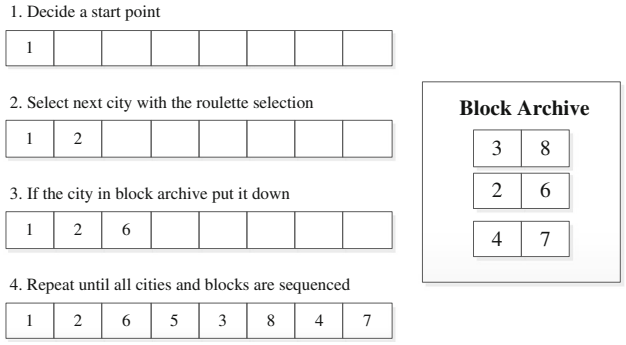

Fig. 7 The approach procedure of the first country

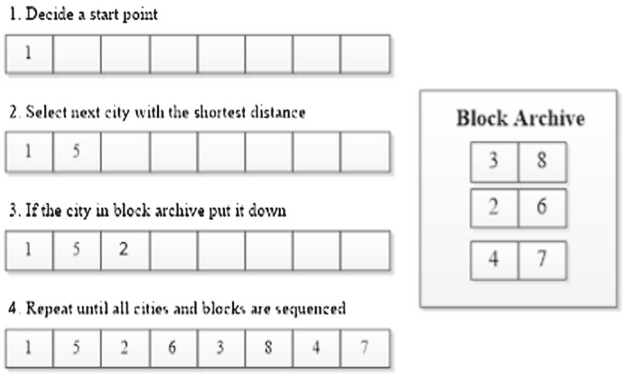

Fig. 8 The approach procedure of the second country

4. The procedures are repeated until a predefined number for the country's population is met.

As shown in Fig. 8, the following proposed procedure is used for generating the second kingdom:

1. A city is picked randomly as the start point.
2. The city with the shortest distance from the head is selected from the residual cities. When the selected city is contained in a block, the block, instead of the selected city, is selected and placed in this chromosome. The last city of the block serves as the new head.
3. Step 2 is repeated until the number of residual cities is zero.
4. The procedures are repeated until a predefined number for the country's population is met.

As shown in Fig. 9, the following proposed procedure is used for generating the third kingdom:

1. A city is picked randomly as the head. A new block archive contains blocks randomly selected from the existing block archive.
2. A city connected with the head is selected from the residual cities by using roulette-wheel selection. When the selected city is contained in a block from the new block archive, the block is selected and placed in this chromosome. The last city of the block serves as a new head.
3. Step 2 is repeated until the number of residual cities is zero.
4. The procedures are repeated until a predefined number for the country's population is met.

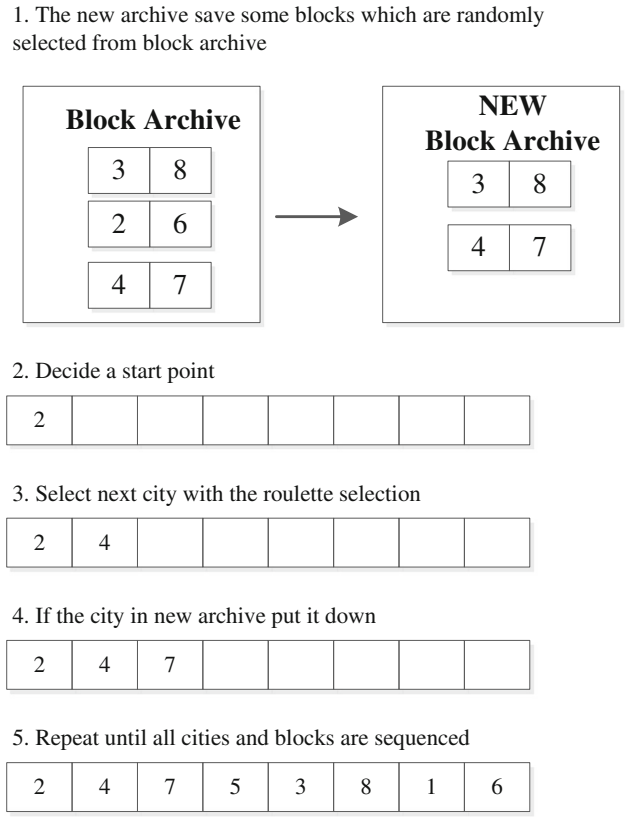

Fig. 9 The approach procedure of the third country

As shown in Fig. 10, the following proposed procedure is used for generating the fourth kingdom:

1. A city is picked randomly as the head. A new block archive contains blocks randomly selected from the existing block archive.
2. The city having the shortest distance from the head is selected from the residual cities. When the selected city is contained in a block from the new block archive, the block is selected and placed into this chromosome. The last city of the block serves as a new head.
3. Step 2 is repeated until the number of residual cities is zero.
4. The procedures are repeated until a predefined number for the country's population is met.

As mentioned, the strategies of blocks applied and choosing the city connected are both two approaches. In blocks-

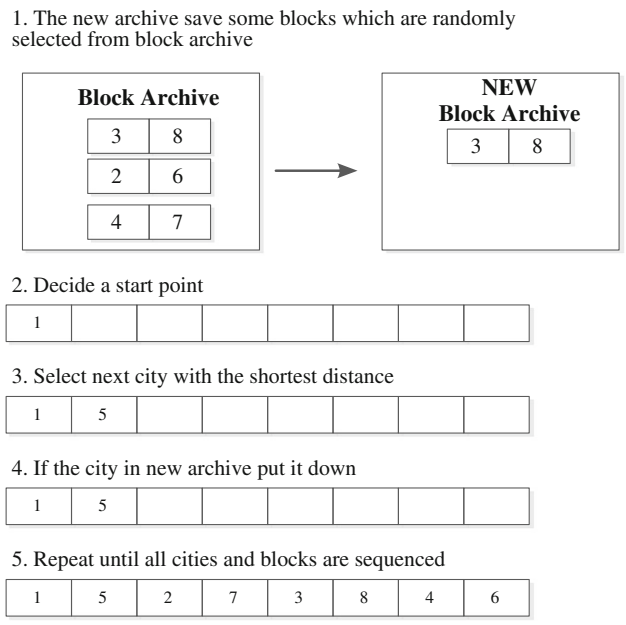

Fig. 10 The approach procedure of the fourth country

applied strategy, one is using all blocks, and the other is just using some of all blocks randomly. In choosing the city connected strategy, one is to choose the city by roulette selection, and the other is by shortest distance. Therefore, there are four approaches of generating countries in our study. The primary idea is to enhance the diversity of country construction in this procedure using different strategies. In other words, to maintain the diversity of populations is to mine feasible solutions.

As shown in Fig. 11, maintaining a continually strong country and increasing its population are the core concepts of the ICA. The BBICA was designed to continually revise each country's population number by calculating the average fitness of each country, ranking each country by average fitness, and then revising each country's population number according to its rank.

Finally, on the basis of the ranking derived from the average-fitness calculations, we use different mutation rules. For this reason, we prefer that weak countries use the powerful mutation rule, which causes the weak countries to improve. However, strong countries use the mutation rule with high diversity to maintain the high diversity of the country's population.

3.4 Local search strategies

In this study, we use two local search strategies, namely the "swap" and the "inverse".

As shown in Fig. 12, the swap chooses two points and then exchanges the two cities. In the TSP, this swap results in the solution construction having four new links.

As shown in Fig. 13, the inverse also selects two points. Subsequently, the inverse causes all of the cities between the chosen points to invert. Specifically, it would exchange City 5 and City 3. The inverse procedure results in the solution construction having two new links in the TSP. The solutions can invoke the method most appropriate for them, depending on what they require to evolve.

4 Experimental results

In this section, we present the experimental results of the BBICA and compare its performance levels with those of other algorithms. Each algorithm was executed 30 times in each instance, and the computing hardware consisted of an Intel Core processor (3.40 GHz) with DDR2 800 memory (2 GB). The programming language used was Microsoft

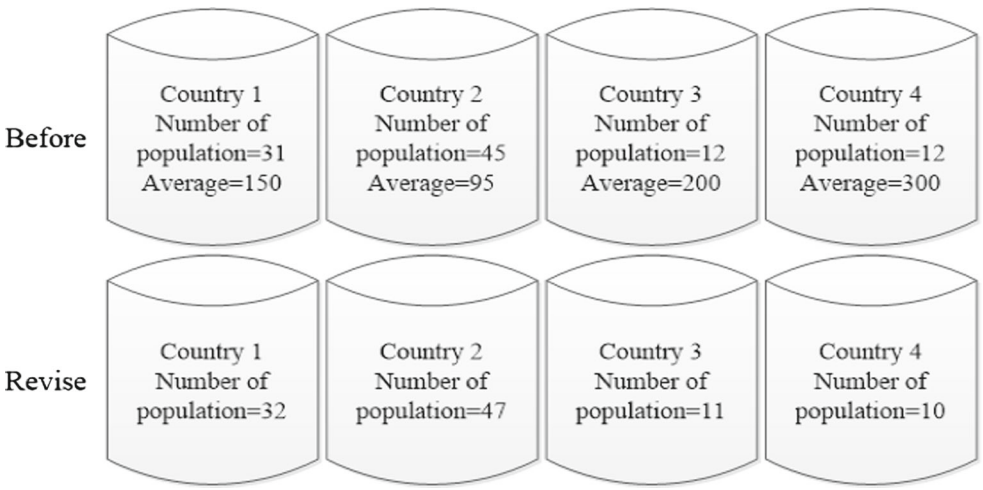

Fig. 11 Revise number of countries population

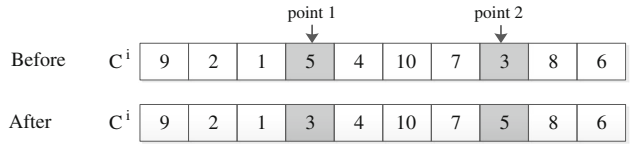

Fig. 12 Swap procedure

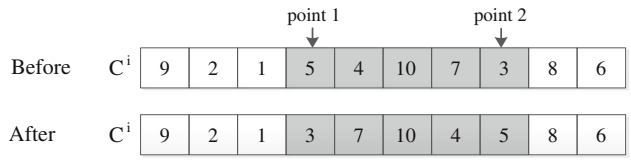

Fig. 13 Inverse procedure

Visual C# 2010 Express. The detailed setup and results of the experiments are explained in the remainder of Sect. 4.

The error ratio (ER) is used to evaluate the solution quality and is determined as follows:

$$\mathrm{ER}(\%) = \frac{\text{observed value} - \text{optimal value}}{\text{optimal value}} \times 100\,(\%), \tag{12}$$

where the observed value is the fitness of the optimal solution from one algorithm and the optimal value is the best known solution for the used instance. According to the ER result of 30 individual runs, we can obtain the minimal ER (MER) and average ER (AER).

4.1 Parameter setting

There are five important parameters of our proposed approach to be decided. After preliminary experimental analysis, parameters were established, as shown in Table 1.

The block size denotes the length of the mining blocks. The block counts are the total number of blocks to be mined after reaching the final number of generations, which is equal to 0.3 times the total number of cities. Finally, the total number of generation is set to equal 50 times the total number of cities.

Table 1 Set of parameters

Parameter	Values of parameter
Number of the initial solutions	100
Number of the elite solutions	20
Block size	2
Block counts	City × 0.3
Generation	City × 50

4.2 Comparison results

All tested instances were chosen from the solutions on the website of the TSP library that are known to be the most effective solutions. Three approaches (i.e., BBEA, RABNET-TSP, and SME) were selected from the TSP library for comparison with our proposed approach. The BBEA is block-based with superior performance levels. The RABNET and SME algorithms are based on approaches employing self-organized map networks and have extremely efficient and effective performance levels. Next, three objectives were used to show the performance of our proposed approach.

4.2.1 Performance of different mutant strategies

BBICA$^{\mathrm{Hybrid}}$ includes hybrid swap and inverse strategies and it is designed to further enhanced the performance of BBICA in solving the TSP. Hence, we evaluated different strategies by comparing the BBICA's AER performance levels for three types, namely BBICA$^{\mathrm{Hybird}}$, BBICA$^{\mathrm{Swap}}$, and BBICA$^{\mathrm{Inverse}}$.

As shown in Table 2, the "Opt" column is the best known result and BBICA$^{\mathrm{Hybird}}$ is a mix of the one-point swap and inverse; BBICA$^{\mathrm{Swap}}$ uses the one-point swap,

Table 2 Performance comparison of TSP's instance

Instance	Opt	Strategies		
		BBICA$^{\mathrm{Hybird}}$ AER (%)	BBICA$^{\mathrm{Swap}}$ AER (%)	BBICA$^{\mathrm{Inverse}}$ AER (%)
eil51	426	1.64	1.47	1.34
eil76	538	2.49	1.37	2.00
eil101	629	3.53	3.28	3.51
berlin52	7542	0.03	0.03	0.03
bier127	118,282	0.87	0.77	1.08
ch130	6110	2.50	2.64	2.93
ch150	6528	0.55	0.59	0.78
rd100	7910	0.86	0.66	0.52
lin105	14,379	0.66	0.11	0.39
lin318	42,029	4.75	6.89	5.13
kroA100	21,282	0.05	0.06	0.38
kroA150	26,524	1.77	2.51	2.05
kroA200	29,368	1.60	2.65	2.08
kroB100	22,141	0.78	0.58	0.86
kroB150	26,130	1.84	2.85	2.18
kroB200	29,437	3.20	5.22	4.04
kroC100	20,749	0.76	0.34	0.37
kroD100	21,294	2.13	3.63	2.30
kroE100	22,068	0.94	1.75	0.97
rat575	6773	5.48	10.28	6.65
rat783	8806	6.11	12.62	8.83
Average		2.03	2.87	2.30

Table 3 Performance comparison of tsp's instance

Instance	Opt	BBICA^{Hybird} MER (%)	BBEA MER (%)	RABNET MER (%)	SME MER (%)
eil51	426	0.94	0.47	0.23	1.64
eil76	538	1.86	1.12	0.56	2.60
eil101	629	2.70	2.07	1.43	1.75
berlin52	7542	0.03	0.03	0.00	2.29
bier127	118,282	0.50	7.32	0.58	1.32
ch130	6110	1.55	1.11	0.57	1.52
ch150	6528	0.41	0.32	1.13	1.58
rd100	7910	0.05	1.18	0.91	1.49
lin105	14,379	0.02	0.02	0.00	0.00
lin318	42,029	3.46	2.95	1.92	2.68
kroA100	21,282	0.01	0.01	0.24	0.60
kroA150	26,524	1.12	0.94	0.58	1.53
kroA200	29,368	1.14	0.89	0.79	2.64
kroB100	22,141	0.31	0.00	0.91	1.84
kroB150	26,130	0.97	0.23	0.51	0.81
kroB200	29,437	1.55	2.36	0.68	0.90
kroC100	20,749	0.00	0.00	0.80	0.83
kroD100	21,294	0.77	0.00	0.38	0.97
kroE100	22,068	0.23	0.17	1.48	1.41
rat575	6773	4.62	9.67	4.05	4.68
rat783	8806	5.29	7.00	5.00	5.79
Average		1.31	1.80	1.08	1.85

Table 4 Performance comparison of TSP's instance

Instance	Opt	BBICA^{Hybird} AER (%)	BBEA AER (%)	RABNET AER (%)	SME AER (%)
eil51	426	1.64	0.47	2.70	3.43
eil76	538	2.49	1.62	3.40	4.52
eil101	629	3.53	3.32	3.12	4.23
berlin52	7542	0.03	0.03	5.18	6.41
bier127	118,282	0.87	7.76	2.20	2.92
ch130	6110	2.50	1.84	2.82	3.23
ch150	6528	0.55	1.53	3.22	3.42
rd100	7910	0.86	1.18	0.91	1.49
lin105	14,379	0.66	0.02	0.15	0.67
lin318	42,029	4.75	3.90	3.97	4.51
kroA100	21,282	0.05	0.01	1.13	1.57
kroA150	26,524	1.77	1.47	3.14	3.31
kroA200	29,368	1.60	2.26	2.80	3.57
kroB100	22,141	0.78	0.30	2.35	2.17
kroB150	26,130	1.84	0.82	1.92	2.59
kroB200	29,437	3.20	3.24	2.37	2.89
kroC100	20,749	0.76	0.06	1.07	1.93
kroD100	21,294	2.13	0.44	1.89	2.59
kroE100	22,068	0.94	0.33	2.93	2.78
rat575	6773	5.48	10.85	5.06	5.91
rat783	8806	6.11	7.95	6.11	6.60
Average		2.03	2.35	2.78	3.37

and BBICA^{Inverse} uses the inverse. The results showed that BBICA^{Hybird} has higher performance than BBICA^{Swap} and BBICA^{Inverse}. Thus, BBICA mixing the different strategies can increase the search space and enhance the BBICA's exploitation ability.

4.2.2 Comparison with other algorithms in low complexity instances

In this section, the comparisons of the experimental results for the BBEA, RABNET, and SME are presented in Tables 3 and 4.

In Table 3 is presented the MER performance of BBICA^{Hybird}, BBEA, RABNET, and SME. The results show that the MER of the most effective solution in BBICA^{Hybird} is 1.31 %, which is superior to those of the BBEA and SME. However, the MER of the most effective solution in BBICA was not as effective as RABNET. In some instances, the BBICA still had more effective performance levels than the RABNET did.

According to Table 4, the results showed that the AER performance for the BBICA^{Hybird} was 2.03 % more effective than those of BBEA, RABNET, and SME. These results indicate that the steadiness of the BBICA's search solution was the most effective of these four approaches.

Subsequently, we considered the convergence performance of the BBEA and BBICA^{Hybird}, as shown in Fig. 14, where the BBICA^{Hybird} is blue line and BBEA is red line.

The converge-graphs of the BBICA^{Hybird} were compared with the BBEA using the rd100, kaoA200, lin318, and rat783 instances. The red line represents the BBEA and the blue line represents the BBICA^{Hybird}. As shown in Fig. 14, the convergence performance of the BBICA was superior to that of the BBEA.

4.2.3 High complexity instances

In the remainder of Sect. 4, we consider the performance of the BBEA and BBICA for solving TSP instances having high complexities. The results are shown in Table 5.

According to Tables 5 and 6, even in the instance with high complexity, BBICA^{Hybird} still has superior performance to BBEA.

There are more mechanisms to enhance diversity in BBICA, including the two strategies to mine blocks, four strategies to construct the artificial chromosomes, and two mutant strategies to recombine the solutions. Thus, BBICA

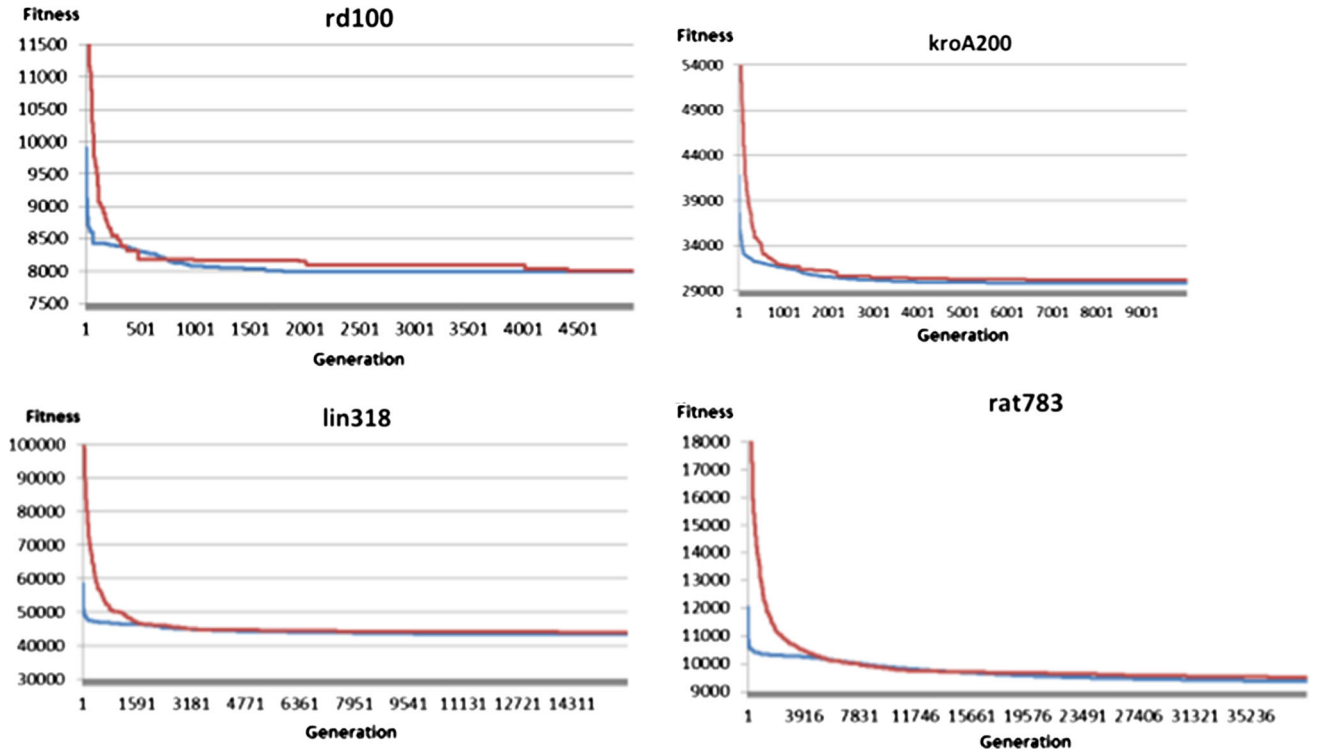

Fig. 14 The convergence graph of BBICA^{Hybird} compared with BBEA

Table 5 Performance comparison of TSP's instance

Instance	Opt	BBICA^{Hybird}		BBEA	
		Best	MER (%)	Best	MER (%)
pr1002	259,045	274,416	5.93	278,040	7.33
pcb1173	56,892	61,142	7.47	61,638	8.34
pr2392	378,032	379,862	0.48	411,449	8.84
Average			4.63		8.17

Table 6 Performance comparison of TSP's instance

Instance	Opt	BBICA^{Hybird}		BBEA	
		Best	MER (%)	Best	MER (%)
pr1002	259,045	275,793.2	6.47	279,270.5	7.81
pcb1173	56,892	61,510.2	8.12	62,066.1	9.09
pr2392	378,032	380,838	0.74	414,710	9.70
Average			5.11		8.87

is more effective than BBEA even though both approaches are block based.

4.2.4 Comparison with the state-of-the-art approach

In this section, the comparison of HMMA (Yong 2015) and our approach are shown in Table 7. The average MER of

Table 7 Performance comparison of TSP's instance

Instance	Opt	BBICA^{Hybird}	HMMA		
		MER (%)	AER (%)	MER (%)	AER (%)
eil76	538	1.86	2.49	2.04	3.72
berlin52	7542	0.03	0.03	0.03	0.03
ch130	6110	1.55	2.50	1.50	3.39
ch150	6528	0.41	0.55	1.94	3.39
lin105	14,379	0.02	0.66	0.11	1.29
lin318	42,029	3.46	4.75	8.12	9.94
kroA100	21,282	0.01	0.05	0.42	0.69
kroA150	26,524	1.12	1.77	2.39	4.99
kroA200	29,368	1.14	1.60	2.15	6.94
kroB100	22,141	0.31	0.78	1.12	2.26
kroB150	26,130	0.97	1.84	1.74	3.22
kroB200	29,437	1.55	3.20	4.89	7.90
kroC100	20,749	0.00	0.76	0.01	1.09
kroD100	21,294	0.77	2.13	0.73	3.28
rat575	6773	4.62	5.48	11.60	14.62
rat783	8806	5.29	6.11	15.25	16.47
Average		1.44	2.03	3.38	5.20

BBICA^{Hybird} is 2.03 % better than HMMA, which demonstrates that the effective search quality of BBICA^{Hybird} is higher than HMMA. In addition, BBICA is also with better

performance in average AER than HMMA, which shows the robust of BBICA^{Hybird} is higher than HMMA.

5 Conclusion

To improve convergence speeds and to avoid being restricted to local optima of the ICA, we proposed a block-based artificial chromosome generation approach for the ICA, namely the BBICA. The BBICA uses the concept of block mining for artificial chromosome generation in the ICA to enhance the fitness quality and maintain the diversity of population. The BBICA exhibits superior performance levels to those of other approaches because the BBICA uses multiple rules to generate blocks that can be further combined into diversified chromosomes with enhanced quality. According to the experimental results, the BBICA has faster convergence speeds than the BBEA, an algorithm that uses only one rule to generate blocks. In addition, our proposed approach uses two local search strategies for increasing its performance levels in solving TSP problems. Finally, comparing the experimental results with those of other well-known approaches validated the concept of the BBICA, an algorithm capable of enhancing the searching ability of the ICA.

In addition, even the result shows that the proposed approach is effective for solving the TSP, but there is some room for improvement in solving high complexity instances. Thus, further work is suggested to develop a mechanism for enhancing the diversity of population to avoid the premature convergence problem.

Compliance with ethical standards

Conflict of interest There is no conflict of interest with this research as known by the authors.

References

Affenzeller M, Wanger S (2003) A self-adaptive model for selective pressure handling within the theory of genetic algorithms. Lect Notes Comput Sci 2809(1):384–393

Atashpaz-Gargari E, Lucas C (2007) Imperialist competitive algorithm: an algorithm for optimization inspired by imperialistic competition. In: IEEE congress on evolutionary computation, pp 4661–4667

Bianchi L, Knowles J, Bowler J (2005) Local search for the probabilistic traveling salesman problem: correction to the 2-p-opt and 1-shift algorithms. Eur J Oper Res 162(1):206–219

Bonyadi RM, Rahimi Azghadi SM, Shah-Hosseini H (2007) Solving travelingsalesman problem using combinational evolutionary algorithm. In: Boukis C, Pnevmatikakis L, Polymenakos L (eds) IFIP international federation for information processing. Artificial intelligence and innovations 2007: from theory to applications, vol 247, Springer, Boston, pp 37–44

Budinich M (1996) A self-organizing neural network for the traveling salesman problem that is competitive with simulated annealing. Neural Comput 8:416–424

Chang PC, Chen MH (2014) A block based estimation of distribution algorithm using bivariate model for scheduling problems. Soft Comput 18(6):1177–1188

Chang PC, Chen SH, Fan CY (2010) Generating artificial chromosomes with probability control in genetic algorithm for machine scheduling problems. Ann Oper Res 180(1):197–211

Chang PC, Chen MH, Tiwari MK, Iquebal AS (2013) A block-based evolutionary algorithm for flow-shop scheduling problem. Appl Soft Comput 13(1):4536–4547

Cheng J, Zhang G, Li Z, Li Y (2012) Multi-objective ant colony optimization based on decomposition for bi-objective traveling salesman problems. Soft Comput 16:597–614

Chu SC, Roddick JF, Pan JS (2004) Ant colony system with communication strategies. Inf Sci 167(1–4):63–76

Goldberg DE (1989) Genetic algorithms in search, optimization and machine learning (book style). Addison-Wesley, Boston

Gutin G, Punnen AP (2002) The traveling salesman problem and its variations. Springer, New York

Holland JH (1973) Genetic algorithms and the optimal allocation of trials. SIAM J Comput 2:88–105

Huang WH, Chang PC, Wang LC (2012) A fast block-based evolutional algorithm for combinatorial problems. World Acad Sci Eng Technol 6:771–777

Johnson DS, McGeoch LA (1997) The traveling salesman problem: a case study in local optimization. In: Aarts EHL, Lenstra JK (eds) Local search in combinatorial optimization. Wiley, Chichester, pp 215–310

Laporte G (1992) The vehicle routing problem: an overview of exact and approximate algorithms. Eur J Oper Res 59(2):345–358

Larranaga P, Kuijpers CMH, Murga RH, Inza I, Dizdarevic S (1999) Genetic algorithms for the travelling salesman problems: a review of representations and operators. Artif Intell Rev 13:129–170

Lee ZJ (2004) A hybrid algorithm applied to traveling salesman problem. In: Proceedings of the 2004 IEEE international conference on networking, sensing and control, pp 237–242

Leung KS, Jin HD, Xu ZB (2004) An expanding self-organizing neural network for the traveling salesman problem. Neural comput 62:267–292

Liu G, He Y, Fang Y, Qiu Y (2003) A novel adaptive search strategy of intensification and diversification in tabu search. In: Proceedings of IEEE international conference on neural networks and signal processing, Nanjing, pp 14–17

Mohammadian M, Sarker R, Yao X (2002) Evolutionary optimization. Kluwer Academic, Boston

Nozarian S, Jahan MV (2012) A novel memetic algorithm with imperialist competition as local search. IPCSIT Hong Kong Conf 30:54–59

Onwubolu GC, Clerc M (2004) Optimal path for automated drilling operations by a new heuristic approach using particle swarm optimization. Int J Prod Res 44(3):473–491

Ouaarab A, Ahiod B, Yang XS (2015) Random-key cuckoo search for the travelling salesman problem. Soft Comput 19(4):1099–1106

Pan QK, Ruiz R (2012) An estimation of distribution algorithm for lot-streaming flow shop problems with setup times. Omega 40(2):166–180

Pasti R, de Castro LN (2006) A neuro-immune network for solving the traveling salesman problem. In: Proceedings of international joint conference on neural networks, vol 6, Sheraton Vancouver Wall Centre Hotel, Vancouver, 16–21 July 2006, pp 3760–3766

Pham DT, Karaboga D (2000) Intelligent optimization techniques: genetic algorithms. Tabu search, simulated annealing and neural networks. Springer, London

Somhom S, Modares A, Enkawa T (1997) A self-organizing model for the travelling salesman problem. J Oper Res Soc 48:919–928

Wang YW, Wu JL, Lin JL (2011) Artificial chromosomes embedded in sub-population genetic algorithm for a multi-objective schedul-

ing problem. In: 3rd international conference on information and financial engineering, vol 12. IPEDR IACSIT Press, Singapore, pp 108–112

Yan XS, Li H, CAI ZH, Kang LS (2005) A fast evolutionary algorithm for combinatorial optimization problem. In: Proceedings of the fourth international conference on machine learning and cybernetics, pp 3288–3292

Yong W (2015) Hybrid max–min ant system with four vertices and three lines inequality for traveling salesman problem. Soft Comput 19:585–596