Chang PC, Huang WH, Zhang ZZ. A puzzle-based genetic algorithm with block mining and recombination heuristic for the traveling salesman problem. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 27(5): 937–949 Sept. 2012. DOI 10.1007/s11390-012-1275-3

A Puzzle-Based Genetic Algorithm with Block Mining and Recombination Heuristic for the Traveling Salesman Problem

Pei-Chann Chang[1] (张百栈), Wei-Hsiu Huang[1] (黄伟修), and Zhen-Zhen Zhang[2] (张真真)

[1]*Department of Information Management, Yuan Ze University, Taoyuan 32026, Taiwan, China*

[2]*Department of Computer Science, Xiamen University, Xiamen 361005, China*

E-mail: {iepchang, weihsiu}@saturn.yzu.edu.tw; zhenzhenzhang222@gmail.com

Received August 24, 2011; revised January 5, 2012.

Abstract In this research, we introduce a new heuristic approach using the concept of ant colony optimization (ACO) to extract patterns from the chromosomes generated by previous generations for solving the generalized traveling salesman problem. The proposed heuristic is composed of two phases. In the first phase the ACO technique is adopted to establish an archive consisting of a set of non-overlapping blocks and of a set of remaining cities (nodes) to be visited. The second phase is a block recombination phase where the set of blocks and the rest of cities are combined to form an artificial chromosome. The generated artificial chromosomes (ACs) will then be injected into a standard genetic algorithm (SGA) to speed up the convergence. The proposed method is called "Puzzle-Based Genetic Algorithm" or "*p*-ACGA". We demonstrate that *p*-ACGA performs very well on all TSPLIB problems, which have been solved to optimality by other researchers. The proposed approach can prevent the early convergence of the genetic algorithm (GA) and lead the algorithm to explore and exploit the search space by taking advantage of the artificial chromosomes.

Keywords artificial chromosome, blocks mining, block recombination, traveling salesman problem

1 Introduction

The traveling salesman problem (TSP), which has been extensively studied by numerous researchers, is known as an NP-hard problem in combinatorial optimization problems (COPs)[1]. It can be stated as follows: Given n cities and the geographical distance between pairs of 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 a search for a permutation, using a cost matrix $\boldsymbol{D} = [d_{ij}]$ to minimize the path length, where d_{ij} denotes the cost of traveling from city i to city j.

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

where π_i denotes the city at the i-th location in the tour.

TSPs can be grouped into different classes based on the properties of the cost matrix. The TSP is symmetric if $d_{ij} = d_{ji}$, $\forall i, j$, asymmetric otherwise; if the nodes lie in a metric space, i.e., distances satisfy the triangle inequality, a problem called the metric TSP is present. Assuming that a city i is marked by its position (x_i, y_i) in the plane, and the cost matrix $\boldsymbol{D}$ contains the Euclidean distance between the i-th and j-th cities, the function is defined as follows:

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

Then, the TSP is both symmetric and metric. The most direct solution would be to try all permutations and see which one has the smallest cost path through all the nodes. The running time for this approach lies within a polynomial factor of $O(n!)$, the factorial of the number of cities, so this solution becomes impractical even for only 50 cities. The TSP was identified by Garey *et al.*[2] to be NP-hard. Many exact and approximation algorithms have been developed for solving TSPs. TSP has attracted the interest of the genetic algorithm (GA) community[3-4] because it is a typical combinatorial optimization problem with wide application in a variety of areas such as vehicle routing, robot control, crystallography, computer wiring, and scheduling.

Recently, GAs have been successfully applied to a wide range of problems including multimodal function optimization, machine learning, COPs as well as the evolution of complex structures such as neural

Regular Paper

networks. An overview of GAs and their implementation in various fields is given by [5]. The advantage of applying GAs to hard combinatorial optimization problems lies in the ability to search the solution space in a broader way than the heuristic methods based upon neighborhood search techniques. Nevertheless, GAs are also frequently faced with a problem of stagnating in a local but not global optimum. This premature convergence of GAs occurs when the population of a GA reaches such a suboptimal state that the genetic operators can no longer produce offspring with a better performance than their parents.

In this research, we introduce a new heuristic using the concept of ant colony optimization (ACO) to extract patterns from the chromosomes generated by previous generations for solving the generalized traveling salesman problem. The proposed heuristic is composed of two phases. In the first blocks mining phase, the ACO technique has been adopted to establish a set of non-overlapping block archive and the remaining cities (nodes) to be visited in set S. The second phase is a block recombination phase where the set of blocks and the rest of the cities are combined to form an artificial chromosome. The generated artificial chromosomes (ACs) then will be injected into the standard genetic algorithm (SGA) process to speed up the convergence. The proposed method is called "Puzzle-based GA" or "p-ACGA".

The paper is organized as follows. In Section 2, the literature regarding GA and TSP is surveyed. In Section 3, the framework and working principle of p-ACGA are introduced. Detailed information for block mining by ACO and block recombination is also provided. Section 4 shows the experimental results and the comparisons of p-ACGA with other algorithms. Section 5 concludes the paper and outlines the areas for future research.

2 Literature Review on GA Approach for TSP

The traveling salesman problem is a classical problem of combinatorial optimization in the area of Operations Research. There are several practical uses for this problem, such as vehicle routing[6] and drilling problems[7]. TSP has been extensively used as a comparison basis in order to improve different optimization techniques, such as genetic algorithms[8], simulated annealing[9], tabu search[10], local search[11], ant colony[12], and neural networks[13].

On the other hand, common, problem-independent heuristics like simulated annealing (SA)[14] and GAs[15-17] deliver poor performance on large TSP instances[18]. They require high execution time for solutions whose quality is often not comparable with those achieved in much less time by their domain-specific local search counterparts. Therefore, a large number of approaches have been developed for solving TSPs. A very promising direction is the genetic algorithm combined with problem specific operators which is named as a memetic algorithm (MA)[19]. MAs adopt the strategy of encoding the population and the genetic operations, so as to direct the individuals' heuristic study and searching direction. The technique does not ensure an optimal solution, however it usually gives good approximations in a reasonable amount of time.

To further improve the GAs for TSPs, many approaches have been proposed. Among these approaches, designing TSPs-specific operators, incorporating domain-specific local searches, and keeping population diversity are considered as promising strategies. Designing TSPs-specified crossovers, such as cycle crossover[20], edge recombination crossover[21], maximally preserving crossover[22], edge assembly crossover[23] and inver-over operators[24], could improve the performances of GAs for solving TSPs. Incorporating domain-specific local search techniques into GAs[25-27] possess both the global optimality of the GAs as well as the convergence of the local search. A new hybrid algorithm was described in [28] that exploits a compact genetic algorithm in order to generate high-quality tours, which are then refined by means of the Lin-Kernighan (LK) local search. The approach showed that keeping population diversity is useful to avoid premature convergences.

Several issues concerning improving the solution quality of GAs for solving TSPs were discussed. Tsai *et al.*[29] addressed these issues by analyzing the behaviors of some crossover and mutation operators on some well-known TSPs. They found that genetic algorithms for TSPs should at least have two mechanisms: powerful genetic operators which can preserve and add "good edges" (i.e., the edges in the near-optimal tour) as well as a mechanism to keep the population diversity.

Multidisciplinary work undertakes to mathematically model these processes and to develop statistical analyses and mathematical algorithms to understand the scrambling in the chromosomes of two or more related genomes[4]. Earlier approaches using "building block" concepts or schemata theory can be referred to messy GAs[30] and fast messy GAs[31]. Messy GAs (mGAs) are a class of iterative optimization algorithms that make use of a local search template, adaptive representation, and a decision theoretic sampling strategy. The work on mGAs was initiated in 1990 by Goldberg *et al.*[30] to eliminate some major problems of the standard GA. In [31], Goldberg *et al.* addressed a major deficiency of mGAs, the initialization bottleneck. During the past decade, mGAs have been applied successfully to a number of problem domains, including permutation-based problems[32]. In general, mGAs evolve a single population of building blocks. A population of building blocks of a predefined length m is initialized during each iteration. A thresholding selection operator is applied to increase the number of fitter building blocks, while discarding those with poor fitness. Then, cut and slice operators are applied to construct global solutions by combining good building blocks together. The best solution obtained is kept as the competitive template for the next iteration. In the next iteration, the length of the building blocks is increased by one (i.e., set to $m+1$). The mGA can be run level by level (i.e., m by m) until a good-enough solution is obtained or the algorithm may be terminated after a given stop criterion is met.

A different approach similar to mGA is proposed in [33]. The authors present a novel co-evolutionary algorithm, the Puzzle Algorithm, where a population of building blocks (BBs) coevolves alongside a population of solutions. They show that the addition of a building-block population to a standard evolutionary algorithm results in notably improved performance on the hard shortest common superstring (SCS) problem. However, due to the nature of BBs that are dependent on the problems and the encoding of the chromosome, their behaviors are difficult to analyze. General problem-independent GAs such as mGA[30] and the Puzzle algorithm[33] are not very efficient in solving TSPs, especially for large problems.

3 *p*-ACGA: A Puzzle-Based Genetic Algorithm

During the evolution process of a GA, each gene of all the chromosomes will slowly converge to a specific position in the chromosomes after many generations. Thus, near the end of the evolution process, the available chromosomes have low diversity and high homogeneity due to the slow convergence. Diverse chromosomes are required to find the global optima.

In our earlier researches[34-36], Artificial Chromosome Genetic Algorithm (ACGA) has been very successful in injecting ACs into the evolutionary process of GA to speed up the convergence. The basic idea behind the insertion of artificial chromosomes is to introduce some diversity among the homogeneous chromosomes. An artificial chromosome (AC) generating mechanism can be developed by collecting the gene information in the archive. The frequency of the occurrence of each city i in position j is calculated by the AC generating mechanism and the frequency is transformed into a matrix called dominance matrix (represented by $\boldsymbol{M}$). This matrix will be further converted into a probabilistic matrix to guide the assignment of each gene to its respective position. These chromosomes can help to locate more efficiently the global optima.

To further improve the solution quality of ACGA, in this research, p-ACGA, a puzzle-based genetic algorithm will be introduced. p-ACGA contains several special characteristics. Firstly, the new method uses ACO pheromone update rules to extract the blocks from the population generated in the previous generations. It is a process of linkage learning which is applied to discover the hidden knowledge within the dependent variables. The common subsequences among a set of highly fit chromosomes are called "blocks". The block consists of a series of genes linked to each other continuously. Secondly, a recombination procedure is adopted to regroup the blocks and the rest of the cities together to generate a set of artificial chromosomes. These ACs are built with very good subsequences (or, micro-structures) and they will be injected into the evolutionary process to speed up the convergence process. The block mining by ACO pheromone update is very effective and can identify a good subsequence from the chromosome. These blocks can reduce the size of the search space, so that the search process may take less time to find the near-optimal solution.

In p-ACGA, an SGA will be adopted and the traditional GA consists of several operators: crossover, mutation and selection, which are responsible for exploitation, exploration and preservation. The population control includes population $\mu+\lambda$ during the evolutionary processes. However, in our proposed approach we will have a different population strategy to include β chromosomes as another source of population generation during the AC injection process. In most applications, GAs can find excellent solutions even the optimal ones. However, in the applications of high complexity GAs are usually trapped in the local optima. p-ACGA aims at mining the linkage information of genes to assemble the strong joining blocks and recombine the mined blocks with the remaining genes to generate ACs to enhance the solutions structure. The framework of the heuristic is depicted as Fig.1, where *Maxgen* means the maximum number of generations.

3.1 Genetic Algorithm

A simple GA is applied to solve the TSP. The process includes the following aspects.

Chromosome Coding. In this paper, we will use the most direct way to denote TSP-path presentation. For example, path 4-2-1-3-4 can be denoted as (4, 2, 1, 3) or (2, 1, 3, 4) and it is referred to as a chromosome. Every chromosome is regarded as a valid path.

Initialization. A set of cities, i.e., n, are generated randomly.

Fitness Value. It is the standard of judging whether

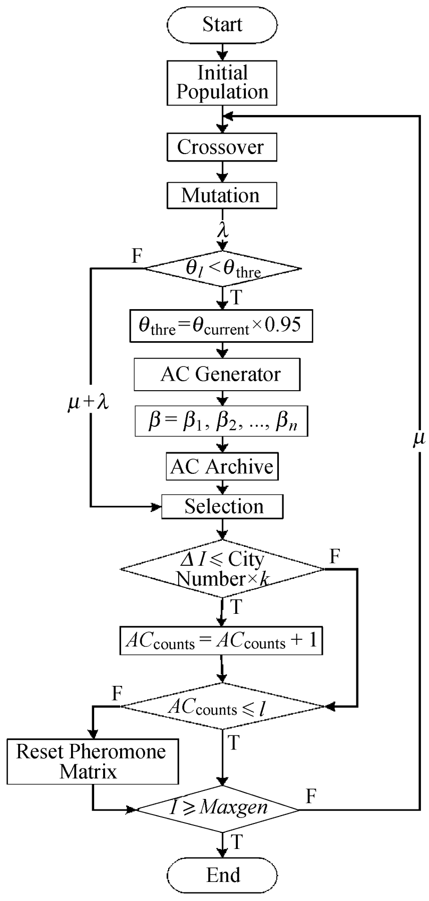


Fig.1. Framework of p-ACGA.

an individual is "good" or not. We take the reciprocal of the length of each path as the fitness function. The shorter the length, the better the fitness values. The fitness function is defined in the following formula:

$$F(\pi)=\frac{1}{f(\pi,d)}=\frac{1}{\sum_{i=0}^{n-1} d_{\pi_i,\pi_{(i+1)}}+d_{\pi_n,\pi_0}}. \tag{3}$$

Selection Strategy. After crossover or mutation, new generations are produced. A truncation strategy is adopted to keep the total number of chromosomes consistent. The tournament selection strategy is applied in this research.

Crossover Strategy. The crossover operator involves the swapping of genetic material between the two parent strings. Two parents produce two offspring. In this research, two different crossover operators, i.e., OX and GSX are applied to generate offspring simultaneously.

Mutation Strategy. The mutation strategy is used to avoid getting trapped into local optima. It aims to gradually generate individuals with better fitness. Four different mutation operators, i.e., Swap, Inverse, PMX and Insert are applied to generate offspring simultaneously.

Terminal Condition. The algorithm will stop once the total number of generations is reached.

After the initialization stage, $\mu+\lambda$ are introduced to the mating pool for selection. An AC injection controlling threshold $\theta_{\rm I}$ is used to decide whether the convergence starts to slow down. If the mechanism is activated, the AC generator will generate new ACs, i.e., β, to be injected into the evolutionary process. Then, these β artificial chromosomes will be combined with population $\mu+\lambda$ to form a new population which will be introduced into the mating pool. Furthermore, another variable k is applied to control the number of times to inject β ACs into the mating pool. If the injection of AC cannot increase $\theta_{\rm I}$ to surpass $\theta_{\rm thre}$, the injection of AC will continue until the count reaches k. In addition, $\tan^{-1}\delta f/\delta I$ is applied to evaluate the fitness differences δf between specific evolving iterations δI. $\theta_{\rm I}$ is designed to dynamically adjust the convergence pressure and effectively inject the artificial chromosomes into the evolutionary process. By injecting ACs into the GA process, a different breed of species is introduced and these species contain a very good microstructure identified by the ACO algorithm. These ACs maintain the high quality blocks which can speed up the convergence speed quickly. At the same time, even the fitness is not compatible after long evolutionary runs, these ACs consisting of high quality of blocks can lead the genetic operator to search for promising solution areas. $AC_{\rm counts}$ is the controller for adjusting the continuous iterations for injecting AC. Once $AC_{\rm counts}$ is greater than l which is a preset number, the pheromone matrix will be reset to $\mathbf{0}$. The population strategy includes β and $\mu+\lambda$. That is the new population μ_{i+1} which will replace the original population μ_i and become the new population μ for next generation. Fig.2 is a schematic diagram illustrating the reconstruction of the population.

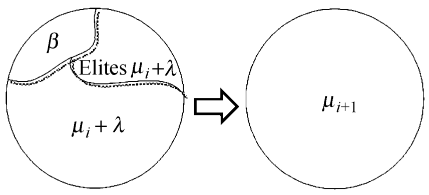


Fig.2. Population reconstruction schematic diagram.

3.2 Definition of Blocks

Data mining[18] is the process of applying clustering, classification and association rule methods to data with the intention of uncovering hidden patterns[37], which will lead the algorithm to recombine the discrete genes more effectively. In this paper, the so-called "Blocks Mining" technique of mining linkage information is applied for joining available segments together as the assembling foundation to recombine the chromosomes. Theoretically, the crossover or mutation of blocks will significantly decrease the complexity of searching solutions when compared with the crossover or mutation of genes. Therefore, the computational time will be reduced while the quality of solutions will be enhanced via recombining the mined blocks.

In this paper, we aim at adopting the mined block to enhance the performance of recombination in AC generation. Basically, block recombination is regarded as the puzzle process, which tries to assemble the puzzle pieces and locate them in the right allocation. Here we propose a "Puzzle-Based Model"(PBM). PBM is formally defined as a triple (H, S, R) where H is a finite set of non-empty blocks (puzzles) with fixed gene length, i.e., 4 here, S is the rest of cities not in the blocks, R is called the recombination function that combines the set of blocks with S together to form a legal chromosomes, i.e., nearest neighbor (NN) in this research. A schematic diagram of the puzzle-based model is illustrated in Fig.3.

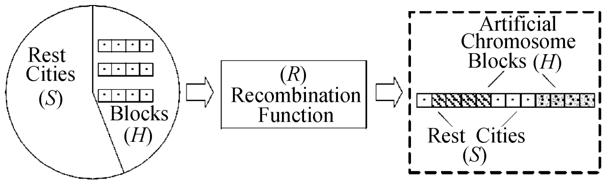


Fig.3. Schematic diagram of the puzzle-based model for AC generation.

Basically, in p-ACGA, we use ACO pheromones intensities to identify a set of suitable blocks which is treated as a small piece of puzzle to be recombined into a legal chromosome. In order to understand the behavior of PBM from the point of view of blocks creation and composition, there is a need for a very simple and direct representation of blocks. In this paper, we propose a new way to look at blocks which are called "puzzles". The proposed definition can be applied directly in several ways both in the identification and the composition process. A block or puzzle can be regarded as a subset of the building block structure. A similar concept can be found in [38, 44]. Here is a 10-city chromosome composed of $3 * 925 * * * 71$ shown in Fig.4. There are three blocks in this schema as follows: $\{3\}$, $\{9, 2, 5\}$ and $\{7, 1\}$ (B_1, B_2 and B_3 respectively).

Many possible patterns of blocks in a chromosome are shown in Fig.5. The minimum size of a block is one allele, i.e., C_3. The maximum size equals to the chromosome length, i.e., C_2. The remaining block sizes are between 1 and $k - 1$, i.e., C_1, where k is the length of the chromosome. The constructed blocks with different patterns will be recombined with the rest of the cities into a chromosome.

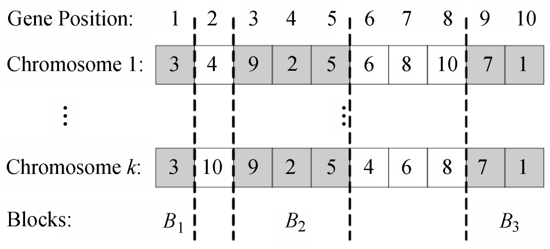

Gene Position: 1 2 3 4 5 6 7 8 9 10
Chromosome 1: 3 4 9 2 5 6 8 10 7 1
Chromosome k: 3 10 9 2 5 4 6 8 7 1
Blocks: B1 B2 B3

Fig.4. Example of block within a schema.

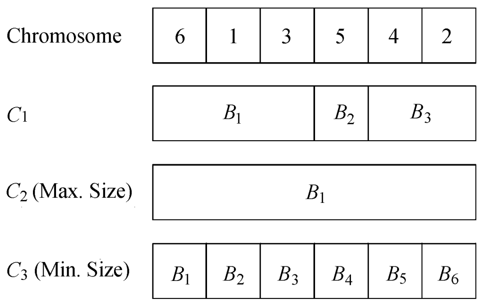


Fig.5. Example of possible patterns of blocks in a chromosome.

3.3 Block Mining by ACO

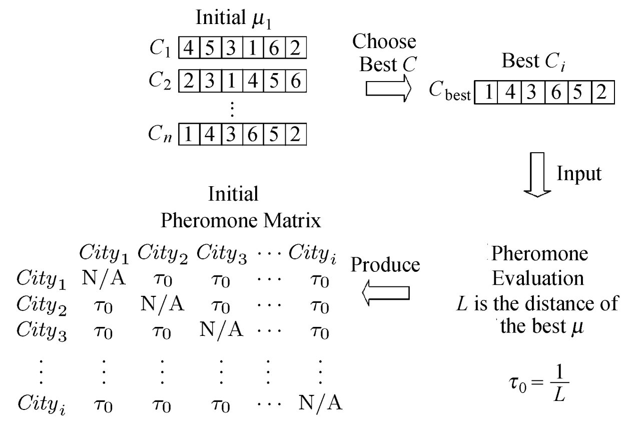
In ACO algorithms, proposed by Dorigo *et al.*[39], simple artificial ants act as co-operative agents to generate high quality solutions to a combinatorial optimization problem via interaction between the ants and their environment. The ants use a stochastic construction heuristic that employs probabilistic decisions on the basis of artificial pheromone trails and problem-specific heuristic information. Pheromone is accumulated during the construction phase through a learning mechanism implied in the algorithm's pheromone update rule. In particular, ants move from one node to another on a construction graph using a transition rule that favors shorter edges and edges with greater amounts of pheromone. They update the pheromone trail of their generated tours based on a pheromone update rule. This rule deposits a quantity of pheromone proportional to the quality (or length) of the corresponding tour.

In GAs, each chromosome holds some information about the solution. We generally accept that good solutions can guide a search method to the desired solution because they contain some useful information. The problem is how to extract such information. The common knowledge between good solutions (the mutual information) can be observed. In the case of two chromosomes, the similarity of bits in the same position is their mutual information. Although there are many different chromosomes that have the same fitness value, there will be some repeat pattern (common knowledge) between them. If the size of population is large enough, this mutual information will be reliable. This increases the chance to find common subsequences and maintains diversity of common patterns.

The common blocks are regarded as having high potential for good substructure or genomes (the building blocks) because they appear identically in two selected chromosomes which are assumed to be good or highly fit. Therefore they will be retained in the original structures.

To identify a block from a set of high-fit chromosomes, in this research pheromone intensity among cities as in ACO will be applied to locate a small piece of block. This block is just like a microstructure within the set of chromosomes. A set of blocks will be mined from these chromosomes and they are similar to fragments of DNA. These blocks will be then recombined with the rest of cities to form an AC. This process is built in an AC generation model. In the AC generation model, there are two major mechanisms applied to produce β number of artificial chromosomes to be injected into the evolutionary process. The first phase is the block mining procedure and the main purpose of this phase is to identify a small set of cities with high percentage of pheromones. This set of cities is called a "block" of the chromosome. A physical chromosome consists of couples of blocks to be identified by ACO algorithm, and is non-overlapping and can be recombined with the rest of the cities later on to form a new legal chromosome.

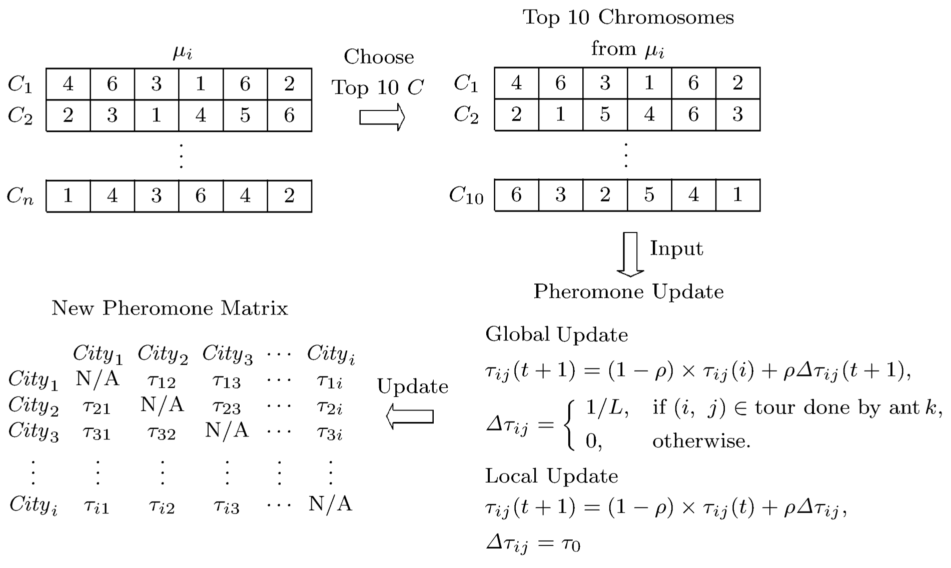
ACO is applied to identify the strength between city to city by the pheromone updating strategy. In the beginning, a pheromone matrix is initialized by selecting μ chromosomes with best fitness from the population in the first generation as shown in Fig.6. The initial pheromone for each city is represented as τ_0 for evaluating the intensities between cities via $\tau_0 = 1/L$, where L denotes the total distance travelled for a complete tour. The data are collected in a matrix to record the pheromone intensities for each city.



Initial μ_1
C_1 4 5 3 1 6 2
C_2 2 3 1 4 5 6
C_n 1 4 3 6 5 2
Choose Best C
Best C_i
C_{best} 1 4 3 6 5 2
Input
Initial Pheromone Matrix
$City_1$ $City_2$ $City_3$ ⋯ $City_i$
$City_1$ N/A τ_0 τ_0 ⋯ τ_0
$City_2$ τ_0 N/A τ_0 ⋯ τ_0
$City_3$ τ_0 τ_0 N/A ⋯ τ_0
$City_i$ τ_0 τ_0 τ_0 ⋯ N/A
Produce
Pheromone Evaluation
L is the distance of the best μ
$\tau_0 = \frac{1}{L}$

Fig.6. Initial pheromone matrix.

Next, the pheromone matrix is updated by population stored in the μ archive. For example, in this research the top 10 best fit chromosomes in μ are selected from the archive as shown in Fig.7. Then compute τ_{ij} of each pair of cities, where ρ represents the evaporation rate[39], and $\tau_{ij}(t+1)$ denotes the updated τ_{ij}. Finally, the pheromone matrix will be updated by these 10 chromosomes.



μ_i
C_1 4 6 3 1 6 2
C_2 2 3 1 4 5 6
C_n 1 4 3 6 4 2
Choose Top 10 C
Top 10 Chromosomes from μ_i
C_1 4 6 3 1 6 2
C_2 2 1 5 4 6 3
C_{10} 6 3 2 5 4 1
Input
Pheromone Update
Global Update
$\tau_{ij}(t+1) = (1-\rho) \times \tau_{ij}(i) + \rho \Delta\tau_{ij}(t+1)$,
$\Delta\tau_{ij} = \begin{cases} 1/L, & \text{if } (i, j) \in \text{tour done by ant } k, \\ 0, & \text{otherwise.} \end{cases}$
Local Update
$\tau_{ij}(t+1) = (1-\rho) \times \tau_{ij}(t) + \rho\Delta\tau_{ij}$,
$\Delta\tau_{ij} = \tau_0$
Update
New Pheromone Matrix
$City_1$ $City_2$ $City_3$ ⋯ $City_i$
$City_1$ N/A τ_{12} τ_{13} ⋯ τ_{1i}
$City_2$ τ_{21} N/A τ_{23} ⋯ τ_{2i}
$City_3$ τ_{31} τ_{32} N/A ⋯ τ_{3i}
$City_i$ τ_{i1} τ_{i2} τ_{i3} ⋯ N/A

Fig.7. New pheromone matrix.

The new updated pheromone matrix is shown in Fig.7. The pheromone intensities τ_{ij} within each pair of cities (i, j) which is represented as the linkage strength between these two cities. This strength can be applied to identify a set of cities with the highest strength as a block.

To convert the pheromone matrix into a probability matrix, (4) is defined as follows. The variable η_{ij} is denoted as the inverse distance of d_{ij} as in [36] to compute the probability P_{ij}[39] for each pair of cities selected. The parameters a and b control the relative importance of the pheromone versus η_{ij} and uns is a set of feasible cities to be visited next.

$$P_{ij} = \frac{\tau_{ij}^{a} \times \eta_{ij}^{b}}{\sum\limits_{j \in uns} \tau_{ij}^{a} \times \eta_{ij}^{b}}, \quad \eta_{ij} = \frac{1}{d_{ij}}. \tag{4}$$

P_{ij} is applied to establish the probability matrix of a set of cities or a complete tour. This probability matrix is then adopted to establish a block using the accumulative probability of a set of cities as shown in Fig.8.

$$\begin{array}{cccccc}
 & City_1 & City_2 & City_3 & \cdots & City_i \\
City_1 & \text{N/A} & P_{12} & P_{13} & \cdots & P_{1i} \\
City_2 & P_{21} & \text{N/A} & P_{23} & \cdots & P_{2i} \\
City_3 & P_{31} & P_{32} & \text{N/A} & \cdots & P_{3i} \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
City_i & P_{i1} & P_{i2} & P_{i3} & \cdots & \text{N/A}
\end{array}$$

Fig.8. Probability matrix.

According to the probability matrix, a city is picked randomly and then the five connected cities with highest probabilities will be branched. For example, city 38 is selected randomly and the next five cities as shown in Fig.9, with the probability in non-descending orders will be 9, 33, 47, 5, and 8. Again, for each city branched it can be further expanded in the next level. For example, city 9 will be selected and further branched for the next 5 cities, until all cities in the same level are branched. The same procedure repeats again and again until it reaches the fourth level. In the final level, there will be 125 cities, i.e., $5 \times 5 \times 5$ cities.

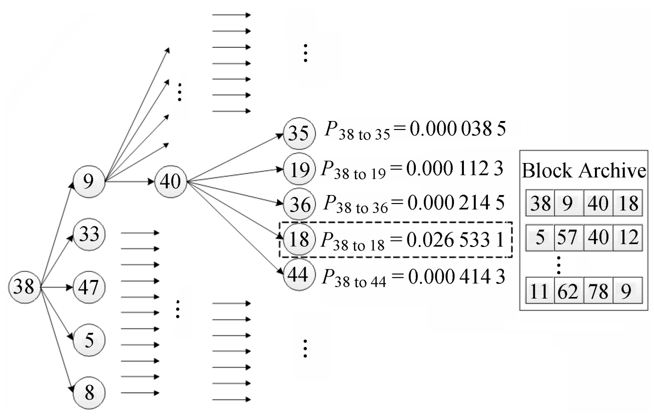

38
9
33
47
5
8
40
35
19
36
18
44
$P_{38\text{ to }35} = 0.000\,038\,5$
$P_{38\text{ to }19} = 0.000\,112\,3$
$P_{38\text{ to }36} = 0.000\,214\,5$
$P_{38\text{ to }18} = 0.026\,533\,1$
$P_{38\text{ to }44} = 0.000\,414\,3$
Block Archive
38 9 40 18
5 57 40 12
11 62 78 9

Fig.9. Set of blocks mined from the probability matrix.

A branching strategy from Branch and Bound (B&B) algorithm[40] for the TSP is applied to generate the possible blocks. In this research, we adopt a forward branching strategy that prescribes which sub-problems should be expanded next and Best First Search (BFS) which solves the most promising sub-problem first, usually the sub-problem with the largest probability value.

A legal block is a set of connecting cities from level 1 up to level 4. In this research, we only branch three times to form a legal block with the length of 4. The reason is to save computational time especially when the problem size is large, i.e., thousand or up to ten thousands of cities. In addition, the block is just like a micro-structure of the chromosome. They can be easily recombined by a heuristic function R in the recombination process to form a longer block.

The probability for each block will be calculated from city to city among these 125 combinations. For example for block {38, 9, 40, 18} the probability is equal to 0.026 533 1. The one with the largest probability will be saved into the block archive. The procedures will be repeated again until a pre-defined number of blocks are identified. If any block with a city overlaps with the blocks previous generated, it will be abandoned. The final set of blocks, i.e., puzzles, are stored in the archive as shown in Fig.9.

3.4 Blocks Recombination

The common sequences are regarded as having high potential for good substructure or the blocks because they appear identically in different selected chromosomes which are assumed to be good or highly fit. Therefore they will be retained in the original structures.

Once the set of blocks are identified and stored in the archive, the rest of cities not in the blocks are also saved together. We name this archive as a puzzle archive as shown in Fig.10.

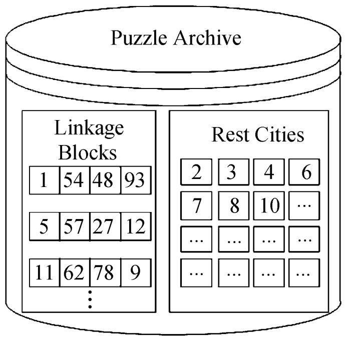


Fig.10. Composition of puzzle archive.

There are numerous methods to recombine these blocks and cities for solving the puzzle to construct a feasible chromosome. The Nearest Neighbor (NN) approach has been applied here in this research. The NN method was initially introduced by Skellam[41] where the ratio of expected and observed mean value of the nearest neighbor distances is used to determine if a dataset is clustered. A diagram by using the NN to form a legal tour is illustrated in Fig.11.

1. Decide the Starting Point

20					

2. Select the City with the Minimal Distance

20	18	$\cdots$	$\cdots$	$\cdots$	

3. Repeat Until all the Cities and Blocks are Sequence

20	18	1	54	48	93

Fig.11. Composition of puzzle archive.

The first step is to randomly choose a city from the set of rest cities as the starting point, then choose a starting city of the block with the minimal distance among these blocks in archive, and repeat until all the cities and blocks are sequenced.

The recombined chromosome here is called AC. AC is the key to maintain the population in GA. Since AC is produced via the block recombination and stored in the AC archive as shown in Fig.12. These β chromosomes will be injected to the population mating pool. These ACs with very good infrastructures can play a paramount role in speeding up the convergence process.

20	19	$\cdots$	$\cdots$	$\cdots$	54
54	33	$\cdots$	$\cdots$	$\cdots$	98
5	9	$\cdots$	$\cdots$	$\cdots$	76
86	44	$\cdots$	$\cdots$	$\cdots$	13

Fig.12. Artificial chromosomes archive.

4 Experimental Studies

The basic parameter setups for GA and ACO are listed in Table 1. Parameters of the basic operators in p-ACGA are exactly the same as those in GA. However, two critical factors in p-ACGA are to be further investigated. The first one is $\varDelta I$ which represents the iteration gaps for injecting the artificial chromosomes. Factor $\varDelta I$ is set to two levels, i.e., 0.5 or 1.0 $\times$ city number (CN). The second factor, AC_{counts}, is designed for the number of times to continuously inject the ACs into the population and it is set to 0.2 and 0.4 $\times$ city number (represented as n). The best combination of these two factors is $\varDelta I = CN \times 0.5$ and $AC_{\text{counts}} = CN \times 0.4$ according to the experimental results.

Table 1. Parameter Setups for GA and ACO

Parameter	Config.	Description
θ_{thre}	0.55	The threshold for controlling the injection of AC, represented as the angle for judging the convergence pressure
$\varDelta I$	$n \times 0.5$ $n \times 1.0$	The total iteration without any update for the found solution, adopted for the AC injection occasion
AC_{counts}	$n \times 0.2$ $n \times 0.4$	The controller for adjusting the continuous iterations for injecting AC
CR	0.8	Crossover threshold is the probability for each chromosome to crossover
MR	0.2	Mutation threshold is the probability for each chromosome to mutate
Selection	$n \times 4$	The parameter used for controlling the selection approach: $\textit{Selection} = \begin{cases} \textit{offsprings}, & \text{if } \varDelta I \text{ is greater than 0}, \\ \textit{Parents} \text{ and } \textit{offsprings}, & \text{otherwise} \end{cases}$
Population	100	Population size
Ants number	30	Number of ants
ρ	0.1	The pheromone evaporation rate
Pheromone update	Global/Local	$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \rho\varDelta\tau_{ij}(\tau_0)$
Pheromone reset	$n \times 5$	The fixed iteration used to reset the ACO for renewing the pheromone matrix
Termination criterion	$n \times 50$	The algorithm will stop once the total number of generations, i.e., $n \times 50$ is reached

4.1 Comparisons with Meta-Heuristics

Since p-ACGA adopts the concept of pheromone update for block mining and the evolving operators of GA, two well-known algorithms, GA and ACO are applied to compare. GA code is from http://www.codeproject.com and ACO is from http://www.aco-metaheuristic.org.

The comparisons of p-ACGA with GA, ACO and ACGA are listed in Table 2, where Opt. means the optimal solution. The CPU time of each algorithm is also listed in Table 3.

The results show that the error rate of GA is relatively higher than the other compared algorithms reaching over 300%. From the plot in Fig.13, p-ACGA is more effective than the other algorithms in instances obtained from the website of TSPLIB (http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/).

Table 2. Comparisons of p-ACGA with Other Approaches

		TSP Instances									
		kroA100	kroA150	kroA200	pr299	pcb442	pr1002	pcb1173	pr2392	pcb3038	Avg.
Opt.		21 282	26 524	29 368	48 191	50 778	259 045	56 892	378 032	137 694	
p-ACGA	Mean	21 547.9	27 302.9	30 467.3	50 019.1	53 719.8	277 906.7	61 184.0	420 618.7	151 321.3	
	Best	21 282	26 714	29 794	48 995	52 263	274 828	60 910	417 300	150 535	
	Std.	256.45	344.57	378.86	654.97	607.47	2 853.23	299.66	5 166.24	801.93	
	Error Rate (%)	1.25	2.94	3.74	3.79	5.79	7.28	7.54	11.27	9.90	5.94
ACO	Mean	26 577.2	33 860.0	39 927.0	66 697.5	68 547.5	360 559.1	78 336.1	539 686.2	245 724.2	
	Best	26 019	32 470	39 058	63 795	65 741	353 632	76 086	530 915	242 969	
	Std.	520.25	924.58	601.79	1 392.03	1 734.50	4 764.69	1 185.04	7 023.87	2 698.83	
	Error Rate (%)	24.88	27.66	35.95	38.40	34.99	39.19	37.69	42.76	78.46	40.00
GA	Mean	27 230.2	34 770.2	44 577.2	115 648.8	136 426.3	1 229 439	282 746.4	3 418 356	1 282 795.0	
	Best	24 960	31 504	39 609	99 315	125 779	1 193 770	270 264	3 355 720	1 257 040	
	Std.	1 866.39	3 090.91	4 581.08	10 242.23	7 768.42	25 263.57	8 952.91	52 932.60	13 282.15	
	Error Rate (%)	27.95	31.09	51.79	139.98	168.67	374.60	396.99	804.25	831.63	314.11
ACGA	Mean	21 596.7	27 346.6	30 559.1	49 977.9	53 783.8	281 503.0	62 663.0	426 801.2	155 919.1	
	Best	21 346	27 109	30 203	48 984	53 388	280 107	62 435	422 839	253 359	
	Std.	173.10	190.19	233.80	547.37	204.66	1 009.68	222.50	2 041.69	1 247.09	
	Error Rate (%)	1.48	3.10	4.06	3.71	5.92	8.67	10.14	12.90	13.24	7.02

Table 3. CPU Time Comparisons of Each Algorithm

TSP	CPU Time (s)			
Instances	p-ACGA	ACO	GA	ACGA
kroA100	45.0	79.3	40.0	6.0
kroA150	96.2	255.1	88.0	14.0
kroA200	167.7	607.8	155.1	25.0
pr299	372.2	1253.4	332.8	59.9
pcb442	796.6	3328.6	713.8	389.5
pr1002	3865.0	5648.6	3624.8	2894.8
pcb1173	5806.0	15320.3	5374.6	4464.2
pr2392	24754.3	48206.7	22776.4	30667.2
pcb3038	30044.1	67654.5	33510.9	61179.5

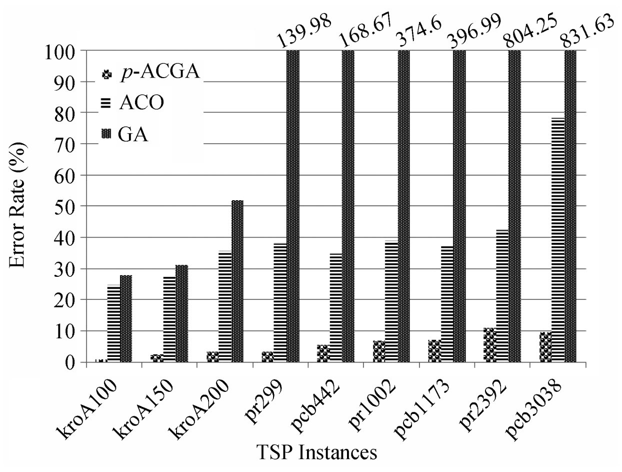


Fig.13. Error rates comparison of the tested parameters.

For the problems with higher complexities, the error rates of the other two algorithms increase significantly. However, p-ACGA still performs well even for large instances with high complexities. In addition, our proposed approach is also very competitive in terms of computational time as shown in Table 3. We did not compare p-ACGA with other effective heuristics such as ILK[45] since p-ACGA did not include any local search heuristic.

4.2 Comparisons with Other Approaches

In this subsection we present the experimental results of p-ACGA and compare the performance of p-ACGA with other algorithms. Each algorithm is executed for 30 times on each instance and the computing hardware consists of Intel Core2 (1.86 GHz) and with DDR2 800 (2 GB Memory). The programming language is Microsoft Visual C++ 2008 Express. All test cases were chosen from website TSPLIB and with the best known solutions.

In order to appropriately tune the algorithms to the problems under investigation, some preliminary experiments were performed and the results are presented in the following. The results to be presented here are the best, mean and standard deviation (Std.) of the cost (tour length) taken over 30 runs.

These two state-of-the-art approaches, i.e., RABNET-TSP[42] and SME[43], are selected for comparison with our proposed approach. These two approaches are based on Self-Organized Map (SOM) network with very effective and efficient performances. The comparisons of the experimental results for p-ACGA, RABNET-TSP, and SME are presented in Table 4, where Opt. means the optimal solution.

Table 4. Comparisons of p-ACGA, RABNET-TSP and SME

TSP Instances	Opt.	p-ACGA				RABNET-TSP				SME			
		Mean	Std.	Best	Error Rate (%)	Mean	Std.	Best	Error Rate(%)	Mean	Std.	Best	Error Rate(%)
eil51	426	430.3	3.8	427	1.00	437.5	4.2	427	2.70	440.6	3.4	433	3.40
eil76	538	548.4	6.1	538	1.90	556.3	5.3	541	3.40	562.3	5.2	552	5.50
eil101	629	641.5	6.9	631	2.00	648.6	3.9	638	3.10	655.6	6.0	640	4.20
beriln52	7 542	7 615.4	126.3	7542	1.00	7 932.5	277.3	7 542	5.20	8 025.1	248.8	7 715	5.80
bier127	118 282	120 377.6	1 616.1	118 695	1.80	120 886.3	1 158.8	118 970	2.20	121 733.3	1 240.0	119 840	3.40
ch130	6 110	6 277.9	60.0	6 137	2.70	6 282.4	60.2	6 145	2.80	6 307.2	63.0	6 203	3.20
ch150	6 528	6 646.6	73.8	6 549	1.80	6 738.4	76.1	6 602	3.20	6 751.1	62.2	6 631	3.90
rd100	7 910	8 044.3	88.8	7 910	1.70	8 199.8	80.8	7 982	3.70	8 239.4	103.9	8 028	4.00
lin105	14 379	14 574.5	170.1	14 379	1.40	14 400.2	44.0	14 379	0.20	14 475.6	118.2	14 379	2.80
lin318	42 029	43 550.1	399.5	42 820	3.60	43 696.8	410.1	42 834	4.00	43 922.9	383.3	43 154	4.20
kroA100	21 282	21 566.5	258.0	21 282	1.30	21 522.7	93.3	21 333	1.10	21 616.8	164.2	21 410	2.60
kroA150	26 524	27 362.4	389.2	26 714	3.20	27 356.0	327.9	26 678	3.10	27 401.3	252.0	26 930	4.60
kroA200	29 368	30 118.8	398.7	29 471	2.60	30 190.3	273.4	29 600	2.80	30 415.7	132.9	30 144	3.70
kroB100	22 141	22 510.1	233.4	22 179	1.70	22 661.5	193.5	22 343	2.40	22 622.5	75.3	25 548	2.90
kroB150	26 130	26 760.7	326.9	26 310	2.40	26 631.9	232.9	26 264	1.90	26 806.3	250.1	26 342	2.80
kroB200	29 437	30 366.0	369.4	29 743	3.20	30 135.0	276.8	29 637	2.40	30 286.5	301.2	29 703	2.80
kroC100	20 749	21 064.6	295.7	20 749	1.50	20 971.2	108.2	20 915	1.10	21 149.9	188.0	20 921	3.20
kroD100	21 294	21 779.1	285.5	21 330	2.30	21 697.4	157.0	21 374	1.90	21 845.7	154.3	21 500	2.30
kroE100	22 068	22 374.6	206.9	22 121	1.40	22 714.6	260.2	22 395	2.90	22 682.5	214.1	22 379	3.40
rat575	6 773	7 152.0	65.7	7 081	5.60	7 115.7	37.5	7 047	5.10	7 173.6	39.5	7 090	5.90
rat783	8 806	9 374.2	80.2	9 235	6.50	9 343.8	47.0	9 246	6.10	9 387.6	39.4	9 316	6.60
rl1323	270 199	296 999.4	2 767.4	294 547	9.90	305 314.3	2 315.8	300 770	13.00	300 899.0	2 717.1	295 780	11.30
fl1400	20 127	21 117.8	58.1	21 037	4.90	21 110.0	163.3	20 851	4.90	20 742.6	115.8	20 558	2.80
d1655	62 128	66 078.4	383.3	65 484	6.40	72 113.2	698.6	70 918	16.10	68 046.4	379.3	67 459	9.50
Avg.					2.99				3.97				4.37

The results show that p-ACGA algorithm outperformed RABNET-TSP[42] and SME[43]. The effectiveness of the proposed approach can be observed in the reduced average error rate for all instances. The results of p-ACGA show that the algorithm is capable of finding the best solution in most cases even for those instances with larger number of cities. Fig.14 shows the error rate of these three approaches on all instances which is computed as (Mean-Opt.)/Opt.×100%.

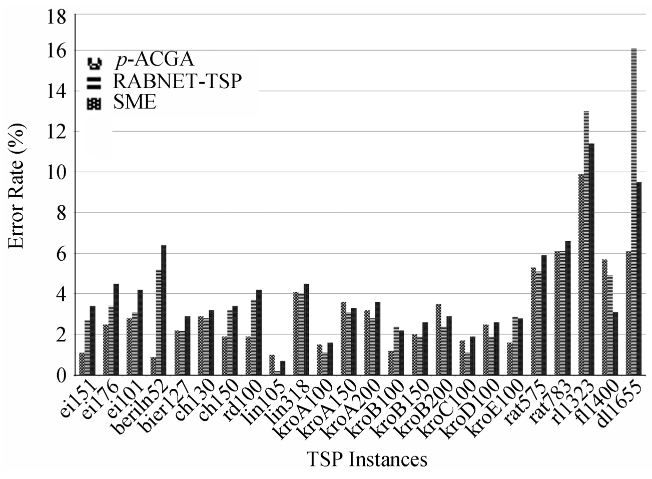


Fig.14. Error rates of these three approaches on all instances.

In addition, the CPU time of p-ACGA are listed in Table 5 for reference. The experimental results show that p-ACGA is very efficient and effective in solving the TSP problems.

For validating the searching ability of p-ACGA on the problems with high complexities, we run p-ACGA with 5 times of computational time which is applied in Table 6. The error rates for these large instances are from 9.9%, 4.9% and 6.4% down to 3.54%, 3.54% and 3.64% for instances rl1323, fl1400 and d1655 respectively. In addition, to verify the significance of the comparisons[46], student t-tests are conducted for these instances. There are two instances, i.e., fl1400 and d1655, that p-ACGA significantly outperforms RABNET-TSP and SME. The other 22 instances are non-significant. The reason is that p-ACGA works pretty much the same as the other two algorithms for small instances. However, p-ACGA performs especially well for large instances.

In addition, two state-of-the-art evolutionary algorithms, i.e., IQCCSA[47] and ESMA[48], are also applied to compare with p-ACGA and the results are shown in Table 7. We leave blanks in the table if the published algorithm did not test on these instances when compared with the proposed p-ACGA. Again, p-ACGA has

Table 5. CPUT Time of p-ACGA

TSP Instances	p-ACGA CPU Time (s)
eil51	48.1
eil76	98.9
eil101	166.9
berlin52	40.7
bier127	246.0
ch130	266.0
ch150	348.3
rd100	162.9
lin105	49.0
lin318	420.3
kroA100	45.0
kroA150	96.2
kroA200	167.7
kroB100	45.0
kroB150	96.3
kroB200	167.9
kroC100	45.0
kroD100	45.0
kroE100	45.0
rat575	1 382.4
rat783	2 558.7
rl1323	7 101.2
fl1400	6 531.4
d1655	8 769.4

Table 6. Experimental Result of p-ACGA Adopting Test Parameters

TSP Instances	Opt.	Mean	Std.	Best	Error Rate (%)
rl1323	270 199	279 765	1 163.5	278 504	3.54
fl1400	20 127	20 840	33.0	20 819	3.54
d1655	62 128	64 390	299.7	64 044	3.64
Avg.					3.57

the minimum mean error rates for most instances.

5 Conclusions

In this study, we presented a block mining and recombination-based genetic algorithm for solving the TSP problems. The blocks are treated like puzzles and the recombination procedures attempt to solve the puzzle to come out with a high fit solution. The quality of the blocks mined from the previous population has a great impact on behavior of p-ACGA. In addition, the blocks can be continuously updated once a new set of elite is identified through the evolutionary process. The length of the block is kept to 4 in this research for these blocks are similar to fragments of DNA. If the block lengths are too big, they may contain redundant information which will be carried over throughout the evolutionary process and the final result may not be good. If the block length is too small, the information contained within each block may be too little and the recombination process may not be able to come out with good quality of chromosomes. The traditional GA is still reserved in p-ACGA since it contains the crossover and mutation operator which will be able to further improve the solution quality of the ACs. ACs are only injected when needed. The reason is that the solution speed can be greatly improved since the mining and recombination process takes some computational time.

Table 7. Comparisons of p-ACGA to IQCCSA and ESMA

TSP Instances	Opt.	p-ACGA		IQCCSA		ESMA	
		Best Err. Rate (%)	Mean Err. Rate (%)	Best Err. Rate (%)	Mean Err. Rate (%)	Best Err. Rate (%)	Mean Err. Rate (%)
eil51	426	0.20	1.10	0.0	1.7		
eil76	538	1.30	2.50	1.3	2.7		
eil101	629	0.20	2.80	3.5	4.6		
st70	675	0.01	0.02	0.4	1.8		
rd100	7 910	0.00	0.02	2.3	3.2		
lin105	14 379	0.00	1.00	0.6	2.1		
pr107	44 303	0.00	0.01	2.4	2.9		
pr124	59 030	0.00	0.01	0.6	1.3		
pr136	96 772	0.02	0.05	6.5	7.4		
pr152	73 682	0.01	0.01	0.5	1.3		
bier127	118 282	0.70	2.20	1.2	2.2		
rat195	2 323	0.02	0.03	2.4	2.9		
kroA200	29 368	1.00	3.20	1.3	2.0	8.2	10.60
kroA100	21 282	0.10	1.50			9.0	10.70
kroA150	26 524	1.30	3.60			4.5	6.40
pr299	48 191	1.70	3.80			7.2	10.20
pcb442	50 778	2.90	5.80			6.0	7.15
pcb1173	56 892	7.10	7.50			15.8	18.10
pcb3038	137 694	9.30	9.90			58.7	59.50
pr1002	259 045	6.10	7.30			13.4	15.20
pr2392	378 032	10.40	11.30			38.2	42.80

The main contribution of this paper is to demonstrate that p-ACGA can be successfully extended to deal with "hard" optimization problems such as the TSP and the effectiveness and efficiency are largely improved when compared with ACO and SGA. Our p-ACGA for the TSP on small problem instances performs very well with approximately 99% solution quality achieved. p-ACGA performed even better when compared with RABNET-TSP and SME on large problem instances. The error rate of p-ACGA is only 2.99% while 3.97% for RABNET-TSP and 4.37% for SME. In summary, we developed a block mining approach by pheromone updating to identify blocks from the elite chromosomes. These blocks can be regarded as small piece of puzzles because they are short, low-order and come from the highly-fit chromosomes. A very effective method is developed in this research to identify these blocks explicitly. Finally, these blocks can be recombined to create better solutions. The experimental results indicate p-ACGA is very effective and efficient. It can be further extended to solve other COPs in the future.

References

[1] Papadimitriou C H, Steglitz K. Combinatorial Optimization: Algorithms and Complexity. India: Dover Publications, 1998.

[2] Garey M R, Graham R L, Johnson D S. Some NP-complete geometric problems. In *Proc. the 8th Annual ACM Symposium on Theory of Computing*, May 1976, pp.10-22.

[3] Ozcan E, Erenturk M. A brief review of memetic algorithms for solving Euclidean 2D traveling salesman problem. In *Proc. the 13th Turkish Symposium on Artificial Intelligence and Neural Networks*, June 2004, pp.99-108.

[4] Sankoff D, Zheng C, Muñoz A, Yang Z, Adam Z, Warren R, Choi V, Zhu Q. Issues in the reconstruction of gene order evolution. *Journal of Computer Science and Technology*, 2010, 25(1): 10-25.

[5] Black T, Fogel D B, Michalewicz Z. Handbook on Evolutionary Computation. USA: Oxford University Press, 1997.

[6] Laporte G. The vehicle routing problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, 1992, 59(3): 345-358.

[7] Onwubolu G C, Clerc M. Optimal path for automated drilling operations by a new heuristic approach using particle swarm optimization. *International Journal of Production Research*, 2004, 42 (3): 473-491.

[8] Affenzeller M, Wanger S. A self-adaptive model for selective pressure handling within the theory of genetic algorithms. In *Proc. the 9th EUROCAST*, February 2003, pp.384-393.

[9] Budinich M. A self-organizing neural network for the traveling salesman problem that is competitive with simulated annealing. *Neural Computation*, 1996, 8(2): 416-424.

[10] Liu G, He Y, Fang Y, Oiu Y. A novel adaptive search strategy of intensification and diversification in tabu search. In *Proc. International Conference on Neural Networks and Signal Processing*, December 2003, pp.428-431.

[11] Bianchi L, Knowles J, Bowler N. Local search for the probabilistic traveling salesman problem: Correction to the 2-p-opt and 1-shift algorithms. *European Journal of Operational Research*, 2005, 162(1): 206-219.

[12] Chu S C, Roddick J F, Pan J S. Ant colony system with communication strategies. *Information Sciences*, 2004, 167 (1-4): 63-76.

[13] Leung K S, Jin H D, Xu Z B. An expanding self-organizing neural network for the traveling salesman problem. *Neurocomputing*, 2004, 62: 267-292.

[14] Kirkpatrick S, Gelatt Jr. C D, Vecchi M P. Optimization by simulated annealing. *Science*, 1983, 220(4598): 671-680.

[15] Grefenstette J, Gopal R, Rosimaita B, Gucht D V. Genetic algorithms for the traveling salesman problem. In *Proc. Int. Conf. Genetics Algorithms and Their Applications*, October 1985, pp.160-168.

[16] Braun H. On solving traveling salesman problems by genetic algorithm. In *Lecture Notes in Computer Science 496*, Schwefel H P, Männer R (eds.), Springer-Verlag, pp.129-133.

[17] Michalewicz Z. Genetic Algorithms + Data Structures = Evolution Programs (3rd edition). Berlin, Germany: Springer-Verlag, 1996.

[18] Fan J, Li D. An overview of data mining and knowledge discovery. *Journal of Computer Science and Technology*, 1998, 13(4): 348-368.

[19] Moscato P, Norman M G. A memetic approach for the traveling salesman problem implementation of a computational ecology for combinatorial optimization on message-passing systems. In *Proc. International Conference on Parallel Computing and Transputer Application*, Sept. 1992, pp.177-186.

[20] Oliver I M, Smith D J, Holland J R C. A study of permutation crossovers on the TSP. In *Proc. the 2nd International Conference on Genetic Algorithm and Their Applications*, July 1987, pp.224-230.

[21] Whitely L D, Starkweather T, Fuquay D'A. Scheduling problems and traveling salesman: The genetic edge recombination operator. In *Proc. the 3rd International Conference on Genetic Algorithms*, June 1988, pp.133-140.

[22] Mühlenbein H, Gorges-Schleuter M, Krämer O. Evolution algorithms in combinatorial optimization. *Parallel Computing*, 1988, 7(1): pp.65-85.

[23] Nagata Y, Kobayashi S. Edge assembly crossover: A high-power genetic algorithm for the travelling salesman problem. In *Proc. the 7th International Conference on Genetic Algorithms*, July 1997, pp.450-457.

[24] Tao G, Michalewicz Z. Inver-over operator for the TSP. In *Proc. the 5th Int. Conf. Parallel Problem Solving from Nature*, September 1998, pp.803-812.

[25] Johnson D S, McGeoch L A. The traveling salesman problem: A case study in local optimization. In *Local Search in Combinatorial Optimization*, Aarts E, Lenstra J K (eds.), John Wiley and Sons, Ltd., 1997, pp.215-310.

[26] Zou P, Zhou Z, Wan Y Y, Chen G L, Gu J. New meta-heuristic for combinatorial optimization problems: Intersection based scaling. *Journal of Computer Science and Technology*, 2004, 19(6): 740-751.

[27] Merz P. A comparison of memetic recombination operators for the traveling salesman problem. In *Proc. the Genetic and Evolutionary Computation Conference*, July 2002, pp.472-479.

[28] Baraglia R, Hidalgo J I, Perego R. A hybrid heuristic for the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, 2001, 5(6): 613-622.

[29] Tsai H K, Yang J M, Tsai Y F, Kao C Y. Some issues of designing genetic algorithms for traveling salesman problems. *Soft Computing*, 2004, 8(10): 689-697.

[30] Goldberg D E, Korb B, Deb K. Messy genetic algorithms: Motivation, analysis, and first results. *Complex Syst.*, 1989, 3(5): 493-530.

[31] Goldberg D E, Deb K, Kargupta H, Hank G. Rapid accurate optimization of difficult problems using fast messy genetic algorithms. In *Proc. the 5th Int. Conf. Genetic Algorithms*, June 1993, pp.56-64.

[32] Kujazew D, Golberg D E. OMEGA-ordering messy GA: Solving permutation problems with the fast messy genetic algorithm and random keys. In *Proc. Genetic Evolutionary Computation Conf.*, July 2000, pp.181-188.

[33] Zaritsky A, Sipper M. The preservation of favored building blocks in the struggle for fitness: The Puzzle Algorithm. *IEEE Transactions on Evolutionary Computation*, 2004, 8(5): 443-455.

[34] Chang P C, Chen S H, Fan C Y. Mining gene structures to inject artificial chromosomes for genetic algorithm in single machine scheduling problems. *Applied Soft Computing*, 2008, 8(1): 767-777.

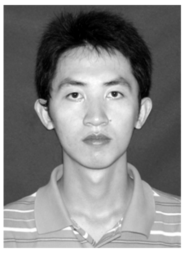
[35] Chang P C, Chen S H, Fan C Y, Chan C L. Genetic algorithm with artificial chromosomes for multi-objective flow shop scheduling problems. *Applied Mathematics and Computation*, 2008, 205(2): 550-561.

[36] Chang P C, Chen S H, Fan C Y, Mani V. Generating artificial chromosomes with probability control in genetic algorithm for machine scheduling problems. *Annals of Operations*

Research, 2010, 180(1): 197-211.
[37] Kantardzic M. Data Mining: Concepts, Models, Methods, and Algorithms. Totowa, USA, Wiley-IEEE Press, 2003.
[38] Sangkavichitr C, Chongstitvatana P. Fragment as a small evidence of the building blocks existence. In *Exploitation of Linkage Learning in Evolutionary Algorithms*, Chen Y P (ed.), Adaptation, Learning, and Optimization, Springer-Verlag Berlin Heidelberg, 2010, pp.5-44.
[39] Dorigo M, Gambardella L M. Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, 1997, 1(1): 53-66.
[40] Narendra P M, Fukunaga K. A branch and bound algorithm for feature subset selection. *IEEE Transactions on Computers*, 1977, C-26(9): 917-922.
[41] Skellam J G. Studies in statistical ecology: I. Spatial pattern. *Biometrica*, 1952, 39(3/4): 346-362.
[42] Pasti R, de Castro L N. A Neuro-immune network for solving the traveling salesman problem. In *Proc. International Joint Conference on Neural Networks*, July 2006, 6: 3760-3766.
[43] Somhom S, Modares A, Enkawa T. A self-organizing model for the travelling salesman problem. *Journal of the Operational Research Society*, 1997, 48: 919-928.
[44] Smith J, Fogarty T C. Recombination strategy adaptation via evolution of gene linkage. In *Proc. the IEEE Conference on Evolutionary Computation*, USA, May 1996, pp.826-831.
[45] Hahsler M, Hornik K. TSP - Infrastructure for the traveling salesperson problem. *Journal of Statistical Software*, 2007, 23(2): 1-21.
[46] Yao X. An empirical study of genetic operators in genetic algorithms. *Microprocessing and Microprogramming*, 1993, 38(1-5): 707-714.
[47] Dai H W, Yang Y, Li C, Shi J, Gao S, Tang Z. Quantum interference crossover-based clonal selection algorithm and its application to travelling salesman problem. *IEICE Trans. Inf. & Syst.*, 2009, E92.D(1): 78-85.
[48] Chang P C, Huang W H, Ting C J. Developing a varietal GA with ESMA strategy for solving the pick and place problem in printed circuit board assembly line. *Journal of Intelligent Manufacturing*, 2010, DOI: 10.1007/s10845-010-0461-9.

Pei-Chann Chang received his M.S. and Ph.D. degrees from the Department of Industrial Engineering of Lehigh University in 1985 and 1989, respectively. He is a chair professor in Yuan Ze University in Taiwan, China. His research interests include fuzzy neural networks, production scheduling, time series data forecasting, evolutionary computation and applications of soft computing. He has published his research work in international journals, such as IEEE Transactions on Systems, Man, and Cybernetics — Part C, Neurocomputing, Decision Support Systems, Applied Soft Computing, European Journal of Operational Research, International Journal of Production Economics, Computers & Industrial Engineering and Computers & Operations Research.

Wei-Hsiu Huang received his M.S. from the Department of Industrial Management in St. John's University, Taiwan, China, 2006 and received his Ph.D. degree from the Department of Industrial Management, Yuan Ze University, China, in 2010. His research interests include Applications of Soft Computing, Production Scheduling, Combinatorial Optimization Problems, and Multi-Objective Optimization Problems. Currently, he is a postdoctoral researcher in Yuan Ze University to develop the meta-heuristic algorithms in combinatorial optimization.



Zhen-Zhen Zhang received his B.E. and M.E. degrees both in computer science from Xiamen University, China, in 2009 and 2012 respectively. He used to be an exchange student for one year in Dept. Information Management from Yuan Ze University, China. He is now a research assistant in Dept. Management Science of City University of Hong Kong. His research interest includes all aspects of computational intelligence, such as scheduling, traveling salesman problem, and vehicle routing.