Discrete Parallel QUasi-Affine TRansformation Evolution Algorithm for Traveling Salesman Problem

Shu-Chuan Chu[1], Xiao-Qi Liu[1], Jeng-Shyang Pan[1,2], Fei-Fei Liu[1], Tien-Szu Pan[3]*

[1] College of Computer Science and Engineering, Shandong University of Science and Technology, China
[2] Department of Information Management, Chaoyang University of Technology, Taiwan
[3] Department of Electronic Engineering, National Kaohsiung University of Science and Technology, Taiwan
scchu0803@sdust.edu.cn, 202282060066@sdust.edu.cn, jspan@cc.kuas.edu.tw,
202082060033@sdust.edu.cn, tpan@cc.kuas.edu.tw

Abstract

The traveling salesman problem (TSP) is a classic combinatorial optimization problem belonging to the NP-hard problem. This paper extends the QUATRE algorithm to this field. The quasi-affine transformation evolution (QUATRE) algorithm provides six evolution schemes and simplifies the setting of control parameters. Because the QUATRE algorithm is easy to fall into local optimal and the convergence performance of the QUATRE algorithm is insufficient for the application, many excellent heuristic algorithms are continuously proposed. This article modified the QUATRE algorithm using reverse learning and mutation strategy. In order to expand the application of the QUATRE algorithm to the TSP problem, a discretization method is adopted to improve the QUATRE. It proposes the discrete parallel quasi-affine transformation evolution (DPQUATRE) algorithm. DPQUATRE beats the comparison algorithm on all 14 test sets of the Traveling Salesman problem library (TSPLIB). TSPLIB is utilized to assess the property of the DPQUATRE algorithm to show the effectiveness of the method. In addition, the error rate metrics PD_{Best} and $PD_{Average}$ are also used to evaluate the performance of the algorithms. The error rate provides a more visual demonstration of the gap between the distance calculated by the algorithm and the shortest distance.

Keywords: DPQUATRE algorithm, Reverse learning and mutation strategy, Parallel, Discretization, Traveling salesman problem

1 Introduction

As the rapid evolution of logistics and transportation industry, the issue of path optimization has captured the interest of many scholars again. The traveling salesman problem (TSP), as one of the most fundamental problems of path optimization and classical operations research, has been widely concerned by relevant researchers. So the research related to the TSP problem has a very far-reaching effect on modern transportation logistics industry. Since the optimal solution is known, the quality of the algorithm can be evaluated [1-2]. The essence of the TSP problem is the minimum consumption problem of crossing all cities and returning to the origin site based on a set of known cities and travel costs [3-5].

The TSP problem was solved in the early days by traditional solutions, such as dynamic programming, the greedy method, and the branch-and-bound method. The dynamic programming method computes the values of all paths first and then generates a graph with weights. The shortest path is chosen each time to move and returns to the initial city after passing through all cities. The greedy method takes a complexity into a series of local optimal choices. Each step is an extension of the current solution until the problem's complete solution is obtained. A typical application of the greedy method is solving optimization problems. The greedy method yields an overall optimal solution for many problems, and even if it does not yield an overall optimal solution, it is usually near the optimum solution. The branch-and-bound method stores all the city nodes and the distance between the nodes through a tree. It begins from the root node and each time the path with the lowest cost is found up until the leaf node.

However, when the quantity of cities is large, the conventional ways are no longer advantageous in terms of time complexity. Therefore many heuristic optimization algorithms are proposed and applied to TSP problems and other engineering optimization problems. Genetic algorithm (GA) proposes a method for solving optimization problems based on Darwinian theory. GA updates new individuals using mutation operators. In recent years, researchers have been proposing new variation operators [6-7]. GA algorithms are often applied to TSP problems due to their high efficiency. And parallelization is often applied to GA to improve the performance of GA algorithms [8-10]. Differential evolution (DE) can also achieve good results in mutation after a lot of time for parameter adjustment [11]. Focused ant colony optimization (FACO) improves on traditional ACO algorithms for multi-node problems, and it dramatically reduces the error from the maximum result [12]. Ant colony optimization (ACO) inspired by the communication between ants, have been proposed to solve complex problems through pheromone communication, and the TSP problem is one of them [13-14]. Particle swarm optimization (PSO) was initially used to solve

*Corresponding Author: Jeng-Shyang Pan; Email: jspan@cc.kuas.edu.tw
DOI: https://doi.org/10.70003/160792642026032702003

continuous type problems, but later scholars proposed many variants of PSO algorithm to solve this problem. For example, swap sequence based PSO (SSPSO) solves the problem that PSO algorithm cannot solve discrete problems [15]. Dynamic transfer functions are proposed to optimize the running time of evolutionary algorithms (EAs) [16]. The seeding growth phase and seed propagation phase are included in tumbleweed optimization algorithms (TOA). TOA improved its global search ability with a multi-group structure [17]. U-shaped and V-shaped diving patterns are used to explore the search space in Gannet optimization algorithms (GOA) [18]. Cat Swarm Optimization (CSO) is derived from the behavior model of cats and consists of two sub-models, namely the tracing mode and the seeking mode [19]. Pigeon-inspired optimization (PIO) was first applied to solve the air robot path planning problems [20]. The PIO algorithm is also used to control the maximum power point tracking (MPPT) technology by improving the Taguchi method [21]. The PIO algorithm has good results in the location distribution problem, with good optimization in selecting the shortest distribution path and the lowest total cost [22]. The PIO algorithm is applied to TSP through the improvement of the Metropolis acceptance criterion of the simulated annealing algorithm. PIO is also applied to Vehicle Routing Problem (VRP) [23-24]. Slime mould algorithms (SMA) have also good results compared with other meta-heuristics, which are derived from the oscillatory mode of slime mould in nature [25-26]. The Fish Migration Optimization (FMO) algorithm is used in the performance study of wireless sensor networks (WSN) [27]. The grey wolf optimization (GWO) algorithm is among the newest heuristic optimization algorithms [28].

Since many algorithms are prone to fall into local optima, there are many strategies proposed by researchers to improve the local optima. The most commonly used strategies are reverse learning, mutation operations, and so on. Parallel strategies can better solve the local optimum problem and speed up convergence. Parallel strategies can have better solution efficiency for problems with large amounts of data [29-30]. Researchers implement parallel strategies based on GA, and we need parallel strategies for group processing when the number of processors exceeds the population size [31]. In order to improve the running efficiency of algorithms in certain memory spaces, many algorithms solve this problem by parallelization [32]. The PSO algorithm is also used for convergence speed optimization by a reverse learning method. The elite particles adopt local-learning behavior [33-34]. PSO often falls into a local optimum, and the local optimum can be improved to some extent by the method opposition-based [35]. The operation of mutation can be not only a fixed mutation rate but also a self-adaptive mutation rate and a deterministically scheduled dynamic mutation rate [36].

The algorithm which is used for modification is the QUATRE algorithm. The Quasi-Affine TRansformation Evolution (QUATRE) algorithm is closely related to Different Evolution (DE), where the property of DE is associated with its parameters and mutation strategy, while the QUATRE algorithm simplifies the control parameters but has better performance [37-38]. The QUATRE algorithm offers six different evolution schemes. And mathematically speaking, the QUATRE algorithm is also very logically powerful. Some engineering optimization problems must be handled by discretizing the data. For example, the discrete GWO algorithm (D-GWO) is used to tackle the TSP problem [39]. In this paper, the QUATRE algorithm is handled by discretization and parallelization. The discrete parallel QUATRE algorithm proposed in this paper can solve the TSP problem well. The main contribution of this paper is the concept of city clusters. The concept of city clusters is proposed to optimize the results of discretization. The main idea of city clustering is to retain the best city sequences while reordering the worse ones.

Section 2 introduces the path distance calculation method for the TSP problem and the basic information of QUATRE algorithm. Section 3 presents the improved DPQUATRE by parallelism and the discretization of the QUATRE algorithm. Section 4 demonstrates the results of comparing on TSPLIB after the discretization. Section 5 shows the conclusions obtained from the experimental results.

2 Related Work

We will present essential information of TSP problem and a simple summary of the QUATRE algorithm.

2.1 Traveling Salesman Problems

TSP is a classical combinatorial optimization problem. The TSP can be described as follows: a salesman has to pass through many cities to sale his merchandise, and this salesman needs to return to his starting place after passing through all the cities from one city. How to seek the shortest path through all the cities and back to the origin city was the key of the problem. The TSP problem can be obtained as shown in Equation (1) below.

$$f(D) = \sum_{i=1}^{n-1} D_{X(i),X(i+1)} + D_{X(1),X(n)} \quad \textbf{(1)}$$

where $X(i)$ means the i-th visited city. $D_{X(i),X(i+1)}$ means the distance between city $X(i)$ to city $X(i+1)$. $D_{X(1),X(n)}$ represents the cost of returning to the starting city after visiting all cities. $f(D)$ means the total expenses of the tsp problem. The benefits anddrawbacks of the algorithm are evaluated by contrasting experimental results with theresults from the optimum solution.

Many excellent algorithms which can be used for TSP problem have been proposed. Since the lower bound of the TSP problem is determined, the quality of the algorithms can be evaluated. As heuristic algorithms have been proposed by researchers, there are many algorithmic solutions that can produce astonishing results in applying them to the TSP problem. However, what we need when evaluating the strengths and weaknesses of algorithms is not a solution full of uncertainties. Rather, we need solutions that have been proven to be good enough to solve the problem.

Since the TSP problem is a typical path optimization problem, we represent the locations of all cities by their

coordinates. Figure 1 illustrates four different ways of choosing a path when the number of cities is 29.

2.2 QUasi-Affine TRansformation Evolution

The evolution mode adopted by the QUATRE algorithm is like the affine transformation of geometry. In geometry (f: X → Y), the following form is adopted: X → MX+B, and the evolution framework adopted is: X → MX+B, its detailed evolutionary framework is as Equation (2): X → MX+B, its detailed evolutionary framework is as Equation (2):

$$X = \bar{M} \otimes \hat{X} + M \otimes B \tag{2}$$

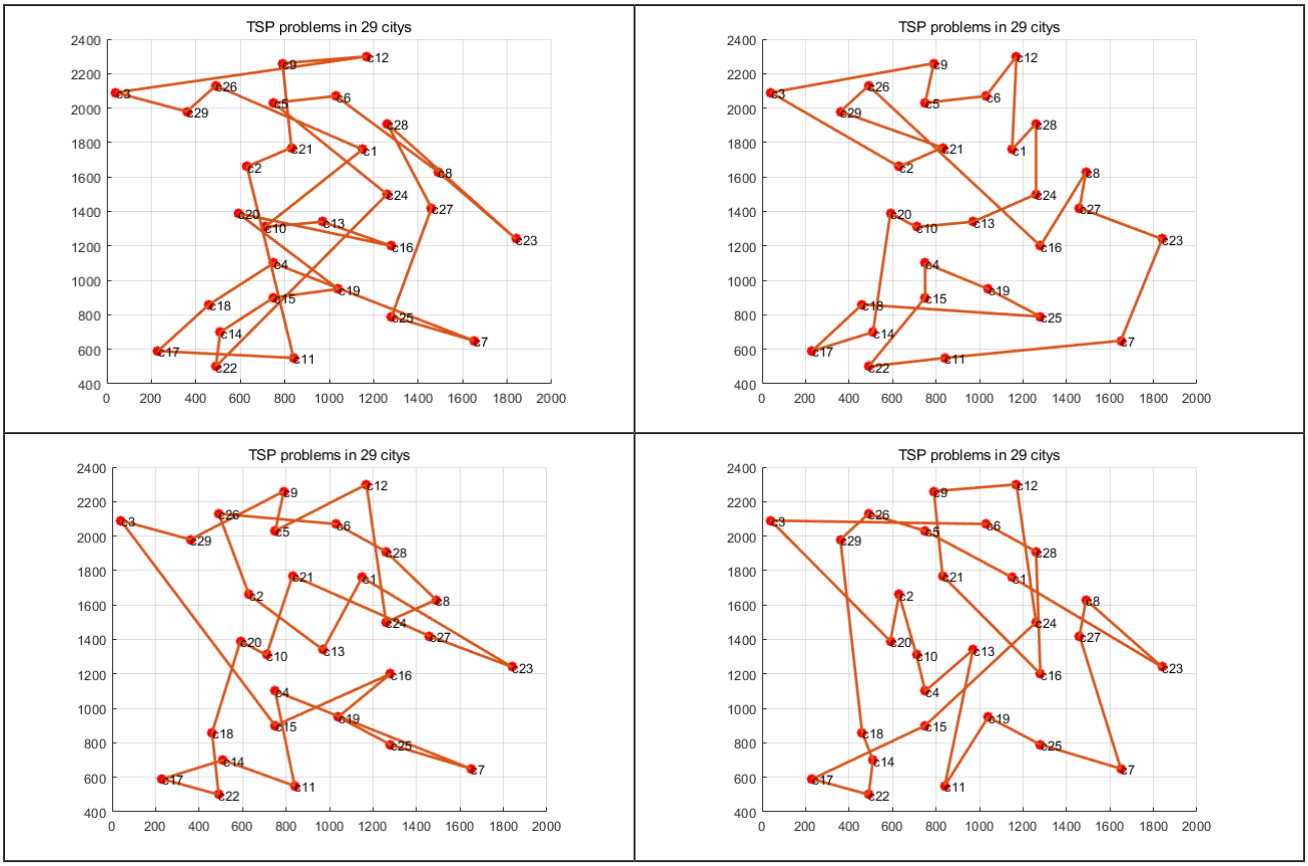


Figure 1. Four different city options under 29 cities

X means the initial matrix, $\hat{X} = (X_1, X_2, \ldots, X_i, \ldots, X_{ps})^T$, $i \in [1, ps]$, ps means the population scale. $\otimes$ is the point multiplication operation, and this operation can be expressed as ". *".

$$M = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} \sim \begin{bmatrix} 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} = \bar{M} \tag{3}$$

M means the co-evolution matrix. $\bar{M}$ means the binary inverse operation on the matrix elements of M. Specific calculation of $\bar{M}$ is shown in the following Equation (3). This step of the procedure is to calculate the evolution matrix, which is an operation on the elements of the matrix. The non-zero elements are replaced by 0 and the zero elements are replaced by 1.

$$M_{\text{int}} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} \sim \begin{bmatrix} 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} \sim \begin{bmatrix} 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} = M \tag{4}$$

P is the number of rows of the matrix and D is the number of columns of the matrix. When $p = D$, each row and column of the initial matrix is reordered randomly. M is the new matrix formed after two transformations. The specific transformation process is shown in Equation (4).

When the number of rows of the matrix is bigger than the number of columns, the number of top-down square matrices is s. k is the number of rows remaining excluding the square matrices. When $p > D$, it can be expressed as $p = s * D + k$ or $p \% D = k$. In this case, each D row is constructed into a lower triangle matrix with lower triangle value of 1, and the lower triangular value of all rows that are not sufficient to form a square matrix is set to 1. The specific conversion process is shown in Equation (5). The

row and column variables are transformed in the same way as Equation (4).

$$M_{\text{int}} = \begin{bmatrix} 1&0&0&0&0\\ 1&1&0&0&0\\ 1&1&1&0&0\\ 1&1&1&1&0\\ 1&1&1&1&1\\ 1&0&0&0&0\\ 1&1&0&0&0\\ 1&1&1&0&0\\ 1&1&1&1&0\\ 1&1&1&1&1\\ 1&0&0&0&0\\ 1&1&0&0&0\\ 1&1&1&0&0 \end{bmatrix} \sim \begin{bmatrix} 1&0&0&0&0\\ 1&1&0&0&0\\ 1&0&0&0&0\\ 1&1&1&1&0\\ 1&1&1&1&1\\ 1&1&1&0&0\\ 1&1&0&0&0\\ 1&1&1&0&0\\ 1&1&1&1&0\\ 1&1&1&1&1\\ 1&0&0&0&0\\ 1&1&0&0&0\\ 1&1&1&0&0 \end{bmatrix} \sim \begin{bmatrix} 1&0&0&0&0\\ 1&0&0&1&0\\ 1&0&1&0&0\\ 1&1&1&1&0\\ 1&1&1&1&1\\ 1&0&0&1&0\\ 1&0&0&1&0\\ 1&0&1&1&0\\ 1&1&1&1&0\\ 1&1&1&1&1\\ 1&0&0&0&0\\ 1&0&0&1&0\\ 1&0&1&1&0 \end{bmatrix} = M \quad \textbf{(5)}$$

Table 1 introduces the calculation method of $B_{i,G}$, and first row in the table is adoptedin this paper. $B_{i,G}$ is the evolutionary guidance matrix, where F is a constant with a value of 0.7. The subscript G of $X_{i,G}$ represents the population of the G generation, $X_{r0,G}$, $X_{r1,G}$, $X_{r2,G}$, $X_{r3,G}$, $X_{r4,G}$ represents new random matrix formed after reordered all the rows of the initial population. $X_{i,G}$ represents the location information of species i. And $X_{gbest,G}$ represents a matrix generated by covering all individuals by optimum individual of the current population. The pseudo-code of the algorithm for QUATRE is displayed in Pseudo 1 below.

3 Discrete Parallel QUasi-Affine TRansformation Evolution Algorithm

Specific schemes for parallel strategies interspecies and intraspecies are presented, and ways to improve the reverse learning and mutation strategies are described. A method to find the shortest path for the TSP problem based on city clusters is proposed.

3.1 Traditional Ranked-order-value Method

This section proposes the discrete parallel quasi-affine transformation evolution algorithm (DPQUATRE). To handle the discretization problem, a heuristic communication strategy is proposed in this paper. This work combines the traditional ranked-order-value method with the city clusters method proposed in this paper to obtain better results. The traditional ranked-order-value method is introduced first.

Algorithm 1. The pseudo-code of the QUATRE algorithm

Input: Explore space [R^D_{min}, R^D_{max}], the maximum number of function calls nfe_{max}, evaluation function formula $f\left(\hat{X}\right)$.

Output: The optimal value $f(X_{gbest,G})$, the position of the optimal solution $X_{gbest,G}$, the number of function calls nfe.

1: Initialize population iteration times $G = 1$, population position information $\hat{X}_G = \left(X_{1,G}, X_{2,G}, \cdots, \cdots, X_{ps,G}\right)^T$.
2: **for** $i = 1$; $i \le ps$; $i++$ **do**
3: Calculate the fitness value $f(X_{i,G})$ of the i-th individual evaluation function.
4: **end for**
5: $nfe = ps$.
6: Mark the optimal value $f(X_{gbest,G})$ and the position $X_{gbest,G}$ of the optimum solution of the population.
7: **While** $\Delta f > eps$ **do**
8: The matrix M is generated according to the Equation(4).
9: **for** $i = 1$; $i \le ps$; $i++$ **do**
10: $B_{i,G} = X_{gbest,G} + F * (X_{r1,G} - X_{r2,G})$.
11: **end for**
12: $\hat{X}_{temp} \leftarrow \bar{M} \otimes X + \mathrm{M} \otimes \mathrm{B}$.
13: The adaptive value of the evaluation function $f\left(\hat{X}_{G+1}\right)$ is calculated.
14: $nfe = nfe + ps$.
15: **for** $i = 1$; $i \le ps$; $i++$ **do**
16: **if** $f(X_{i,G}) > f(X_{i,temp})$ **then**
17: $X_{i,G} = X_{i,temp}$;
18: **end if**
19: The optimal value of population $f(X_{gbest,G})$ and the position of the optimum solution $X_{gbest,G}$;
20: $G = G + 1$.
21: **end for**
22: **end while**
23: Return $f(X_{gbest,G})$, $X_{gbest,G}$ and nfe;

Table 1. Calculation method of $B_{i,G}$

No.	QUATRE	Equation
1	QUATRE/best/1	$B_{i,G} = X_{gbest,G} + F * (X_{r1,G} - X_{r2,G})$
2	QUATRE/rand/1	$B_{i,G} = X_{r0,G} + F * (X_{r1,G} - X_{r2,G})$
3	QUATRE/target/1	$B_{i,G} = X_{i,G} + F * (X_{r1,G} - X_{r2,G})$
4	QUATRE/target-to-rand/1	$B_{i,G} = X_{i,G} + F * (X_{r0,G} - X_{i,G}) + F * (X_{r1,G} - X_{r2,G})$
5	QUATRE/target-to-best/1	$B_{i,G} = X_{i,G} + F * (X_{gbest,G} - X_{i,G}) + F * (X_{r1,G} - X_{r2,G})$
6	QUATRE/target/2	$B_{i,G} = X_{i,G} + F * (X_{r1,G} - X_{r2,G}) + F * (X_{r3,G} - X_{r4,G})$
7	QUATRE/rand/2	$B_{i,G} = X_{r0,G} + F * (X_{r1,G} - X_{r2,G}) + F * (X_{r3,G} - X_{r4,G})$

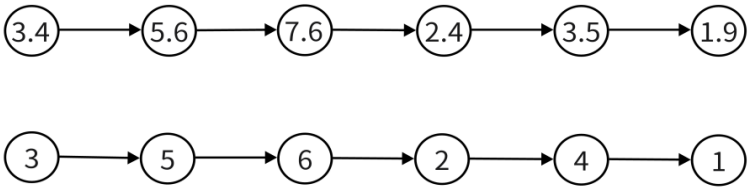


Figure 2. Initial discretization

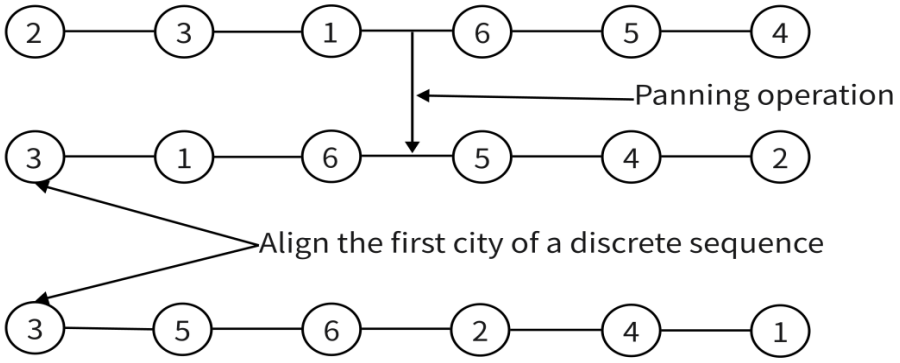


Figure 3. The method of shifting

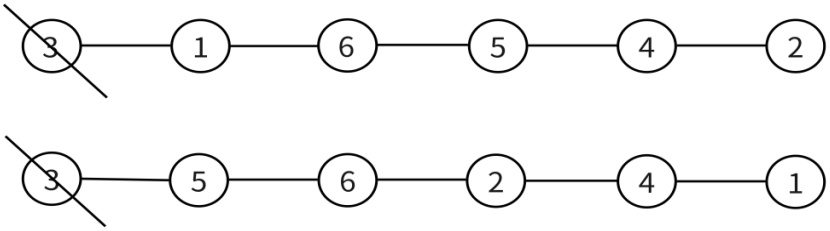


Figure 4. The way to delete a visited city

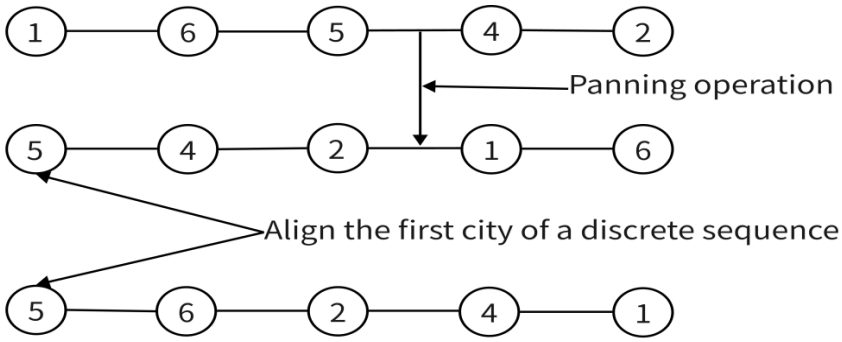


Figure 5. The way to the next city sequence

Each iteration will produce an optimal path. The distance to the next city is compared between the optimal path and another random path and the shorter path is selected. The discretization process is split into two parts. The particle is represented by the first row of vectors.

(1) The initialized particles are sorted in the order of smallest to largest. For example, the array a = [3.4, 5.6, 7.6, 2.4, 3.5, 1.9] is sorted in the order of size as a = [3, 5, 6, 2, 4, 1]. Figure 2 shows the first step of discretization.

(2) This method corresponds the first city of the randomly chosen path to the first city of the optimal path. The specific calculation steps are given in Figure 3. "path 1" in line 1 represents the randomly selected path and "path 2" in line 2 represents the current optimal path.

After obtaining the first city number, calculate the $D_{X(1),X(2)}$ of the two paths separately and remove the city that has been visited. The calculation method is shown in Equation (1). When a city number is obtained, we remove the first city number in order to avoid visiting duplicate cities. The specific steps are shown in Figure 4.

If the distance from city "3" to city "5" is shorter than the distance from city "3" to city "1", then we choose to shift the "path 1". The specific steps are shown in Figure 5.

After several iterations, this method generates a new path. Since this method is similar to dynamic programming, but the distances between cities are all different, the new path generated is might shorter than the current optimum path. Therefore, we generate a new path and compare the new path with the optimal path. If the new path has a shorter distance, we replace the city number of the optimum path with the city number of new path. If the distance of the optimal path is shorter, then we keep the city number of the optimal path.

3.2 Discretization Methods Based on City Clusters

In order to solve the problem that traditional methods are ineffective in finding optimal paths, a method based on city clusters to find more paths is proposed. Optimal city clusters are used to find more optimal paths when cities find a relatively good path. By adopting parallelism and discretization, we propose a new algorithm entitled DPQUATRE algorithm. DPQUATRE has achieved very good results in both results and stability in the TSP problem.

3.2.1 Parallel Intraspecific Strategies

Based on the QUATRE algorithm, a parallel mechanism is added to divide the initialized population into four groups. The concept of city clusters was first applied to the initialization of a discrete sequence of cities. All city clusters consist of a continuous sequence of cities. The sequence of city clusters is shown in Figure 6 below. The number of city clusters is generally chosen to be 1/8 or 1/4 of the total number of cities. The first line of Figure 6 shows the initial sequence and the second line shows the updated sequence. A city cluster consists of a set of continuous sequences, and a complete city sequence contains multiple city clusters. And during each iteration, the distances of all city sequences are calculated. On this basis we can find the optimal city cluster among all city clusters. The best city cluster will be saved and used for the modification of the city sequence. We reorder all cities outside the optimal city cluster to find a more optimal path. If a better path is found, then we replace the original path with it. If the new path is worse, we will keep the original path. The purpose of this initialization step is to find even better paths while preserving the good city sequence as much as possible.

As for intraspecific communication, we are inspired by traditional reverse learning and extend it to the optimization of the worst few city clusters. And the worse city clusters are reordered to find better results. Figure 7 demonstrates the reverse learning method.

Based on the QUATRE algorithm, the following mechanisms are added in this paper: In terms of intraspecies communication strategy, this paper improved the strategy of reverse learning for the worst few city clusters in the group. The process of reverse learning to find better city cluster is shown in the following Equation (6).

$$\begin{cases} X_{best} = X_{ini}, X_{ini} < X_{occ} \\ X_{best} = X_{occ}, X_{ini} \geq X_{occ} \end{cases} \quad \textbf{(6)}$$

X_{best} denotes the sequence of cities selected after initialization, X_{occ} denotes represents the sequence of cities selected after the optimal city cluster operation. X_{ini}

represents the initially generated city sequence. Select the better of these paths as the initialized city sequence.

The intraspecific evolutionary method refers to the idea of reverse learning. Instead of operating on a single point but directly adjusting within the city clusters. After the initialization work is done, we conduct a operation in search of optimal city clusters within the population. After finding the optimal city cluster, the worst city cluster for each population is selected in each iteration of the generation. The reordering of the worst city groups is our main operation in the iterative process. This operation can greatly reduce the generation of large distance cities on a local scale. The reverse learning operation is shown in Figure 7. After the intraspecific communication, we still choose the more optimal path between them. The selection is shown in Equation(7). Where x_i denotes the updated path and X_{best} denotes the path selected after initialization. X_{mbest} represents the new path generated.

$$\begin{cases} X_i = X_{best}, X_{best} < X_{mbest} \\ X_i = X_{mbest}, X_{best} \geq X_{mbest} \end{cases} \quad (7)$$

3.2.2 Parallel Interspecies Strategies

Interspecific communication strategy: The parallel strategy divides all the discretized city sequences into four groups, and the optimization within the groups will often fall into a local optimum. Every 20 iterations, individuals in different groups experience communications with each other. We will do the following for each of the four groups. Each group selects its best sequence of cities and randomly combines it with the best sequences of the other groups. Duplicated cities will appear in the best sequences of other groups, and we remove the duplicated cities and reorder all the cities that have not appeared before. The final new city sequence consists of three parts: the best sequence of the group, the best sequences of the other groups, and the remaining city sequences. Individuals who interact with each other make a judgment. By comparing this city sequence with the original city sequence, we choose the better path as our updated path. The combination of the two optimal city groups in the city sequence and the random combination of the remaining cities usually leads to a much better result. The strategy of mutation is inspired by the optimization algorithm interspecific communication strategy. A similar mutation strategy is adopted for city clusters. The specific operation process is shown in Figure 8. The specific selection methods for mutations are shown in Equations (8) below:

$$\begin{cases} X_i = X_i, X_i < X_{mutate} \\ X_i = X_{mutate}, X_i \geq X_{mutate} \end{cases} \quad (8)$$

Where X_{mutate} donates the individual after mutation and X_i donates the updated individual. If the mutation does not lead to better results, then we still keep the original results. When an individual does not find a better result for a long time, then we can find other paths through the strategy of mutation. The pseudo-code of DPQUATRE is shown in Algorithm 1. The flow chart of the algorithm is shown in Figure 9.

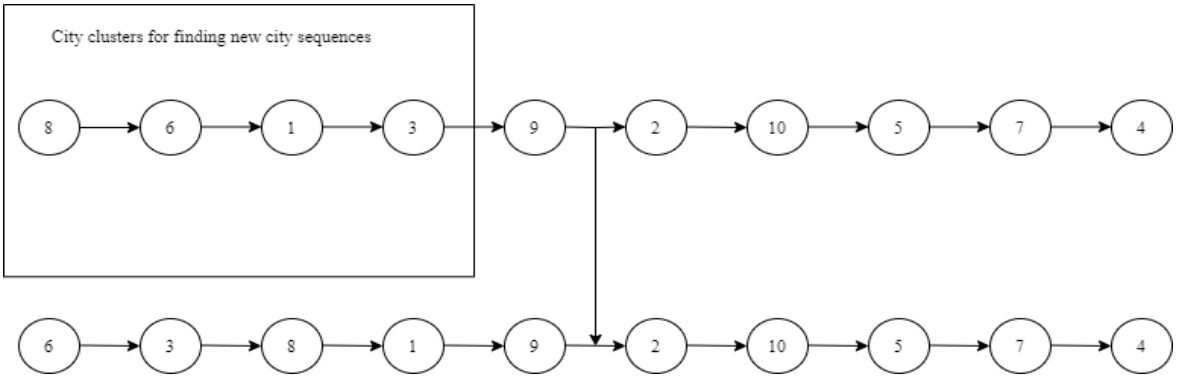


Figure 6. Initialization of urban agglomerations

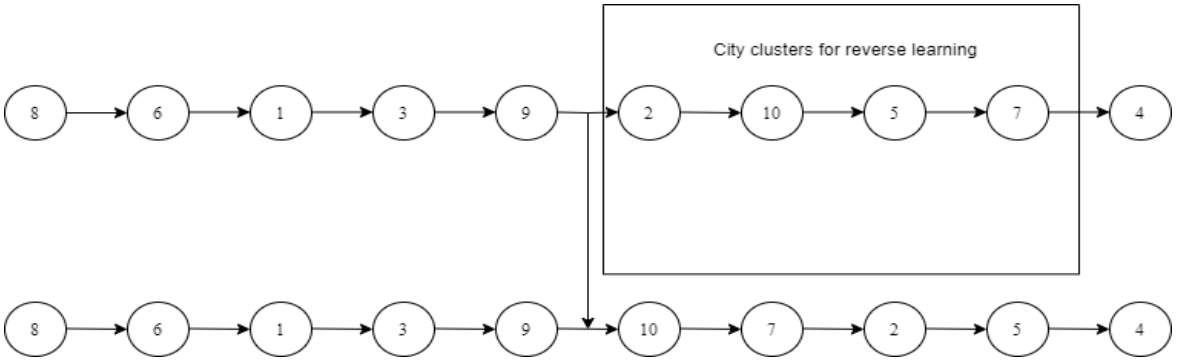


Figure 7. Reverse learning in city clusters

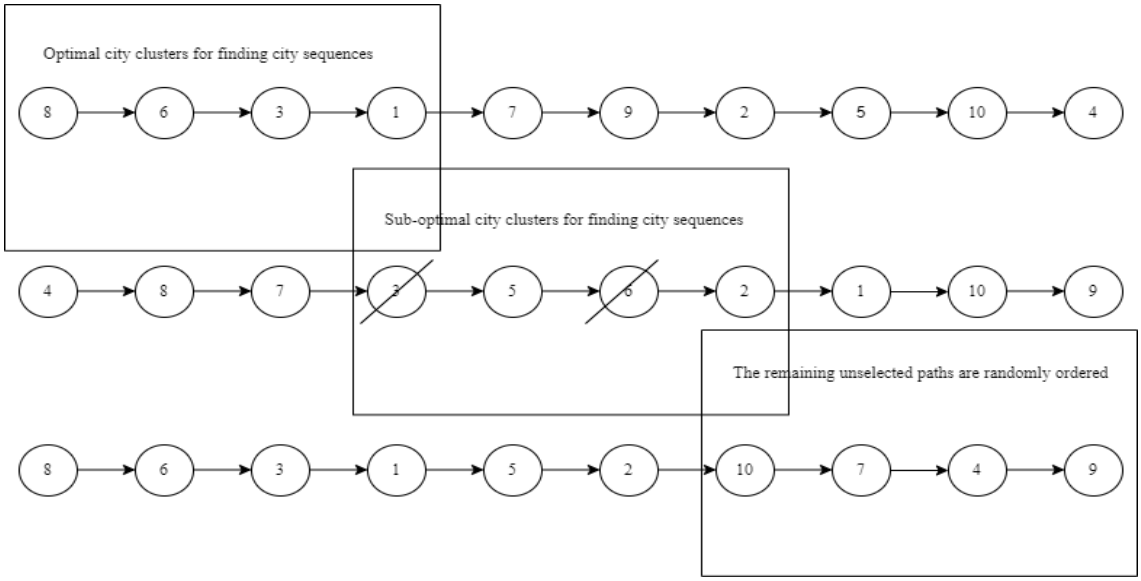


Figure 8. Interspecific communication in city clusters

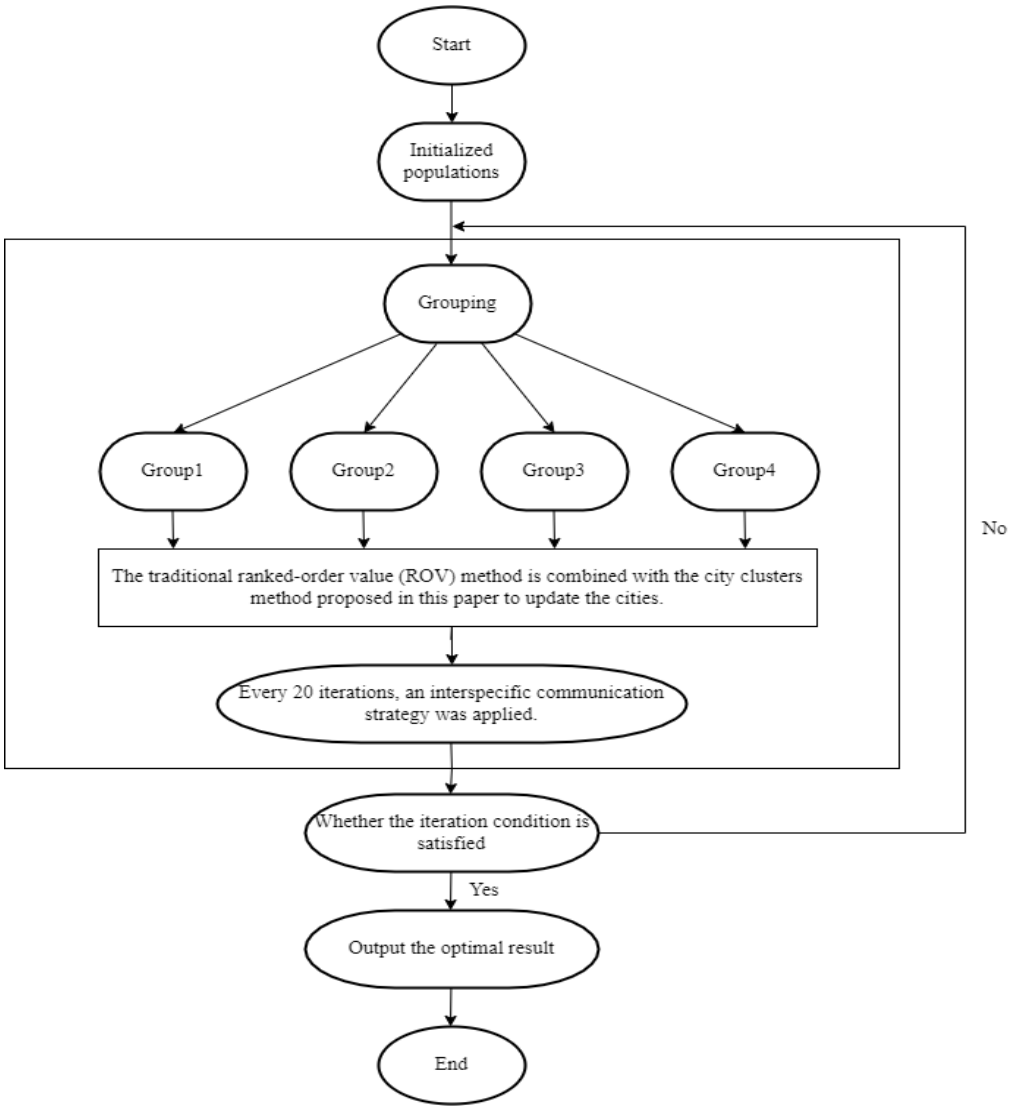


Figure 9. Flow chart of the algorithm

4 Simulation Results

The paper will show the results measured by the DPQUATRE on TSPLIB. The best and worst values are measured by the DPQUATRE algorithm on 14 test sets at TSPLIB. This section will show the results of the comparison with PSO, ACO and GA.

4.1 Experimental Operation Environment

All experimental environments in this paper are the same. The experiments were all done using a personal laptop that is on the windows 10 operating system, 32GB memory, Intel(R) Core(TM) i7-11800H CPU @ 3.00 GHz 3.00 GHz. The software used to perform the experiment is Matlab2021b.

4.2 Parameter Setting

Table 2 shows all the parameter settings for the four algorithms compared. iteration represents the total number of iterations, 1000 iterations for all four algorithms. Pop represents the population size selected by the algorithm which means the total number of city sequences. cc represents the number of groupings, and since the DPQUATRE algorithm was parallelized, it grouped all the populations, while the other three algorithms did not. cs represents the number of populations in each group after parallelization of the DPQUATRE algorithm.

4.3 Test Results on TSPLIB

The traveling salesman problem library (TSPLIB) is used for the comparison of the results of this algorithm [40]. The PSO algorithm has experienced several developments since it was proposed in 1995 [41]. PSO is compared with the DPQUATRE on TSPLIB to evaluate the validity of DPQUATRE [42]. GA solves the TSP problem by splitting the TSP into multiple sub-problems and then solving the sub-problems separately [43]. DPQUATRE is contrasted with the GA after the order crossover and the center inverse mutation method [44]. ACO handles the TSP problem after the local search technique which is compared with the DPQUATRE [45-46].

The test set adopted in this paper is the TSPLIB test set. Table 3 demonstrates the test index of DPQUATRE on 14 test sets. a280 and att532 represent the test set that has been used. Best represents the optimal solution. *BKS* is the shortest path of the current test set. *Average* represents the average value. Worst represents the worst solution. *Std* represents the variance. PD_{Best} represents the multiple of the difference with the shortest path *BKS*. Each test set was tested 10 times to produce the final results. Since PD_{Best} is a multiple of the difference, the smaller PD_{Best} means better. Equation (9) is the calculation of PD_{Best}.

$$PD_{Best} = \frac{(Best - BKS)}{BKS} \tag{9}$$

PD_{Best} works similarly to error rates. In addition this paper tests the stability on 14 TSPLIB test sets by $PD_{Average}$. The $PD_{Average}$ is calculated as shown in Equation (10) below.

$$PD_{Average} = \frac{(Average - BKS)}{BKS} \tag{10}$$

Average represents the average value obtained in ten tests of the same test set. *BKS* represents the optimal solution of the current test set.

All the data are from the optimal results of ten experiments. For a more intuitive understanding of the comparison of the experimental results, "No." means the value of test sets, and "Name" means the name of the selected TSPLIB dataset. The results of 14 test sets can verify the validity and practicality of the DPQUATRE.

Table 4 shows the results of DPQUATRE algorithm in comparison with ACO, GA, PSO on this metric of PD_{Best}. Table 5 shows the results of the DPQUATRE algorithm in comparison with ACO, GA, PSO on the metric of $PD_{Average}$.

In test outcome of TSPLIB, the experimental outcome are closest to the optimal solution of the current test set. The DPQUATRE algorithm outperforms GA, ACO, and PSO algorithms in terms of results.

By comparing *BKS* with Best, we can see that the DPQUATRE algorithm is very close to the optimal solution on the 14 test sets. At the same time the values of Best and Average are very close to each other and the stability of the algorithm can be guaranteed. To visualize the final test results more intuitively, PD_{Best} is used as theerror rate to test the performance of the algorithm. We can visualize that the error rates of the algorithms are mostly below 20%. The error rate can be kept below 10% on two test sets. The difference between the results of $PD_{Average}$ and PD_{Best} is minimal. $PD_{Average}$ also has an error rate of less than 20% on most test set.

Table 2. Values for parameter settings

Related parameters	Parameter setting	Meaning
Iteration	1000	Max value of iterations
Pop	200	The total value of groups
cc	4	The value of groups
cs	50	The value of inter-group communication

Table 3. Test results of the improved algorithm under several metric

Name	BKS	Best	Average	Worst	Std	PD_{Best}	$PD_{Average}$
a280	2579	3.05E+03	3.15E+03	3.22E+05	4.13E+01	1.83E-01	2.21E-01
att532	86729	1.06E+05	1.08E+05	1.09E+05	1.20E+03	2.22E-01	2.45E-01
bier127	118282	1.27E+05	1.30E+05	1.35E+05	2.76E+03	7.66E-02	9.90E-02
ch150	6528	7.28E+03	7.45E+03	7.67E+03	1.32E+02	1.15E-01	1.41E-01
eil101	629	7.24E+02	7.33E+02	7.38E+02	4.79E+00	1.51E-01	1.65E-01
fl417	11861	1.38E+04	1.40E+04	1.41E+04	1.13E+02	1.66E-01	1.80E-01
gil262	2378	2.81E+03	2.88E+03	2.92E+03	3.73E+01	1.82E-01	2.11E-01
kroA200	29368	3.89E+04	3.94E+04	4.05E+04	5.69E+02	3.24E-01	3.42E-01
kroB150	26130	2.98E+04	3.03E+04	3.11E+04	5.15E+02	1.39E-01	1.59E-01
kroC100	20749	2.28E+04	2.37E+04	2.42E+04	4.75E+02	1.01E-01	1.42E-01
kroD100	21294	2.36E+04	2.45E+04	2.51E+04	4.84E+02	1.09E-01	1.50E-01
kroE100	22068	2.43E+04	2.48E+04	2.51E+04	3.12E+02	1.01E-01	1.23E-01
linhp318	41345	4.92E+05	5.08E+05	5.19E+05	9.12E+02	1.90E-01	2.28E-01
ch130	6110	6.68E+03	6.98E+03	7.20E+03	1.79E+02	9.42E-02	1.42E-01

Table 4. Comparative results of different algorithms on PD_{Best}

No.	Name	DPQUATRE	GA	ACO	PSO
1	**a280**	**1.83E-01**	4.39E+00	4.15E+00	4.44E+00
2	**att532**	**2.22E-01**	8.05E+00	7.91E+00	8.20E+00
3	**bier127**	**7.66E-02**	1.11E+00	1.14E+00	1.26E+00
4	**ch150**	**1.15E-01**	2.17E+00	2.20E+00	2.39E+00
5	**eil101**	**1.51E-01**	1.13E+00	1.14E+00	1.29E+00
6	**fl417**	**1.66E-01**	1.22E+01	1.18E+01	1.23E+01
7	**gil262**	**1.82E-01**	3.06E+00	2.96E+00	3.12E+00
8	**kroA200**	**3.60E-01**	2.23E+00	2.17E+00	2.29E+00
9	**kroB150**	**1.39E-01**	1.74E+00	1.78E+00	1.87E+00
10	**kroC100**	**1.01E-01**	1.23E+00	1.33E+00	1.48E+00
11	**kroD100**	**1.09E-01**	9.28E-01	1.06E+00	1.28E+00
12	**kroE100**	**1.01E-01**	9.36E-01	1.08E+00	1.41E+00
13	**linhp318**	**1.90E-01**	4.07E+00	3.87E+00	4.07E+00
14	**ch130**	**9.42E-02**	9.54E-01	1.03E+00	1.16E+00

Table 5. Comparative results of different algorithms on $PD_{Average}$

No.	Name	DPQUATRE	GA	ACO	PSO
1	**a280**	**2.21E-01**	4.48E+00	4.27E+00	4.52E+00
2	**att532**	2.45E-01	8.85E+00	8.01E+00	8.30E+00
3	**bier127**	**9.90E-02**	1.17E+00	1.16E+00	1.29E+00
4	**ch150**	**1.41E-01**	2.38E+00	2.36E+00	2.58E+00
5	**eil101**	**1.65E-01**	1.14E+00	1.16E+00	1.30E+00
6	**fl417**	**1.80E-01**	1.24E+01	1.19E+01	1.26E+01
7	**gil262**	**2.11E-01**	3.11E+00	2.97E+00	3.17E+00
8	**kroA200**	**3.42E-01**	2.27E+00	2.20E+00	2.36E+00
9	**kroB150**	**1.59E-01**	1.81E+00	1.79E+00	1.94E+00
10	**kroC100**	**1.42E-01**	1.26E+00	1.36E+00	1.55E+00
11	**kroD100**	**1.50E-01**	9.56E-01	1.10E+00	1.34E+00
12	**kroE100**	1.23E-01	9.58E-01	1.12E+00	1.48E+00
13	**linhp318**	**2.28E-01**	4.11E+00	3.91E+00	4.13E+00
14	**ch130**	**1.42E-01**	1.09E+00	1.05E+00	1.18E+00

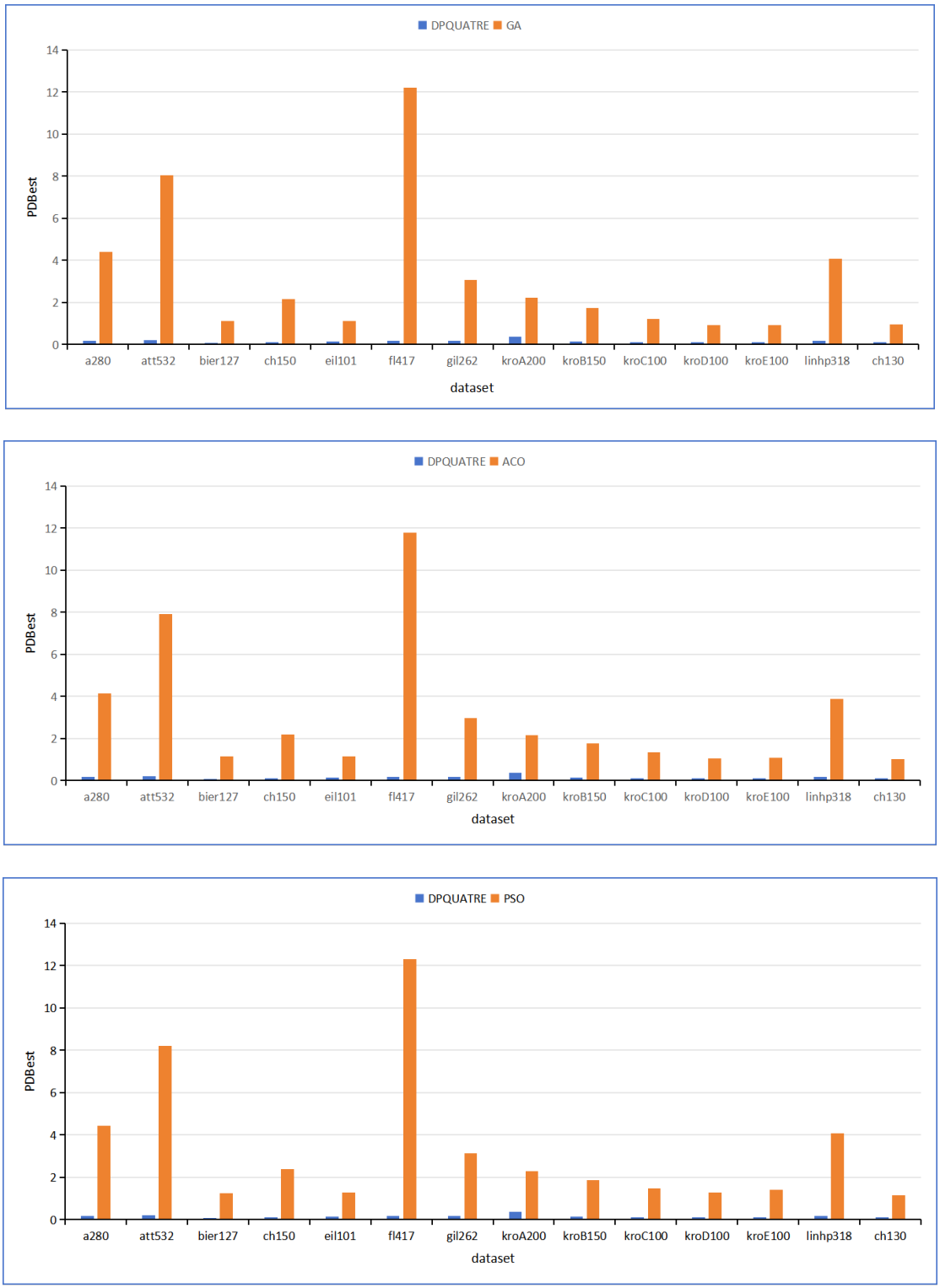


Figure 10. Comparison results of algorithms on PD_{Best}

Comparing DPQUATRE's PD_{Best} on all 14 TSPLIB test sets, DPQUATRE achieved superior results. We use three sets of bar charts to represent how well DPQUATRE is optimized for the PD_{Best} metric relative to the GA, ACO, and PSO algorithms. Figure 10 illustrates specific results. The abscissa of the bar graph represents the name of the selected test set. The blue icon in the bar graph represents the value of the DPQUATRE algorithm and the orange icon represents the values of the other three algorithms.

Comparing DPQUATRE's $PD_{Average}$ across the 14 TSPLIB test sets, DPQUATRE achieves superior results on all of them. However, there are also two problems with this method. Although the algorithm computes substantially better results than the other algorithms, the algorithm computes each set of city clusters at each iteration. As a result, the algorithm takes about ten times as long to compute as the three algorithms. In addition, the choice of size for the city clusters compresses the optimization space for

the remaining cities, so the choice of city clusters also has an impact on the results. In this paper, a relatively good city cluster size is selected after several experiments.

5 Conclusion

We extend the QUATRE to the TSP problem. According to the QUATRE algorithm, we propose methods to discretize it. Limited by the performance of the QUATRE algorithm, this paper proposes city clusters method to improve the performance of the QUATRE algorithm and proposes the DPQUATRE algorithm. And in this paper, the performance of the algorithm is improved by parallel method. The application of mutation and reverse learning further optimizes the performance of the algorithm in the discretization process. For DPQUATRE, we verify the validity by using 14 benchmarks from the TSPLIB, and the experimental outcome demonstrates that DPQUATEE has optimal outcome on all 14 benchmarks. Further investigation will be made into more accurate and speedy intelligent optimization algorithms. Although the discretization performance is improved by the reverse learning strategy, there is still great room for enhancement. Since each operation produces multiple individuals and each individual is required to compute multiple clusters of cities, the average time consumption of DPQUATRE is around 10 times that of the other three algorithms. The choice of the size of the urban clusters can greatly affect the final results, so experimental verification is needed for the choice of the size of the urban clusters. We obtain superior results by combining optimal city clusters with other city sequences. Hopefully we can find a way to seek from one city to another. In future work, we intend to enhance the property of the QUATRE by combining QUATRE with GOA to achieve this wish.

References

[1] M. Jünger, G. Reinelt, G. Rinaldi, The travelling salesman problem, *Handbooks in Operations Research and Management Science*, Vol. 7, pp. 225-330, 1995. https://doi.org/10.1016/S0927-0507(05)80121-5

[2] D. Davendra, *Traveling salesman problem: Theory and applications*, BoD–Books on Demand, 2010.

[3] R. Matai, S. P. Singh, M. L. Mittal, *Traveling salesman problem: an overview of applications, formulations, and solution approaches*, Intech, Crotia, 2010. https://doi.org/10.5772/12909

[4] D. L. Applegate, R. E. Bixby, V. Chvátal, W. J. Cook, *The traveling salesman problem: a computational study*, Princeton university press, 2011.

[5] K. Smith-Miles, J. Van Hemert, X. Y. Lim, Understanding TSP Difficulty by Learning from Evolved Instances, in: C. Blum, R. Battiti (Eds.), *Learning and Intelligent Optimization: 4th International Conference, LION 4, Venice, Italy, January 2010. Selected Papers*, Springer, 2010, pp. 266-280. https://doi.org/10.1007/978-3-642-13800-3_29

[6] E. Alkafaween, A. B. A. Hassanat, Improving TSP Solutions Using GA with a New Hybrid Mutation Based on Knowledge and Randomness, *Komunikácie*, Vol. 22, No. 3, pp. 128-139, 2020. https://www.ceeol.com/search/article-detail?id=1120777

[7] N. Garg, M. K. Kakkar, G. Gupta, J. Singla, Impact of genetic operators on the performance of genetic algorithm (GA) for travelling salesman problem (TSP), *AIP Conference Proceedings*, Vol. 2357, No. 1, Article No. 100020, May, 2022. https://doi.org/10.1063/5.0080965

[8] C. Peng, Parallel genetic algorithm for travelling salesman problem, *International Conference on Automation Control, Algorithm, and Intelligent Bionics (ACAIB 2022)*, Qingdao, China, 2022, pp. 259-267. https://doi.org/10.1117/12.2639457

[9] H.-J. Xu, Y.-S. Ge, G.-D. Zhang, Genetic algorithm for Traveling Salesman Problem, *Proceedings of the 2022 5th International Conference on Computational Intelligence and Intelligent Systems*, Quzhou, China, 2022, pp. 33-40.

[10] A. D. Placido, C. Archetti, C. Cerrone, A genetic algorithm for the close-enough traveling salesman problem with application to solar panels diagnostic reconnaissance, *Computers & Operations Research*, Vol. 145, Article No. 105831, September, 2022. https://doi.org/10.1016/j.cor.2022.105831

[11] R. Mallipeddi, P. N. Suganthan, Q. K. Pan, M. F. Tasgetiren, Differential evolution algorithm with ensemble of parameters and mutation strategies, *Applied Soft Computing*, Vol. 11, No. 2, pp. 1679-1696, March, 2011. https://doi.org/10.1016/j.asoc.2010.04.024

[12] R. Skinderowicz, Improving Ant Colony Optimization efficiency for solving large TSP instances, *Applied Soft Computing*, Vol. 120, Article No. 108653, March, 2022. https://doi.org/10.1016/j.asoc.2022.108653

[13] G.-Q. Chao, S. Aruljodey, O.-C. Xian, A.-S. Bin, Z.-A. Salam, Solution optimization of ACO in TSP by modification of parameters, *Journal of Applied Technology and Innovation*, Vol. 6, No. 1, pp. 58-64, January, 2022. https://doi.org/10.65136/jati.v6i1.159

[14] Y. Wang, Z. Han, Ant colony optimization for traveling salesman problem based on parameters optimization, *Applied Soft Computing*, Vol. 107, Article No. 107439, August, 2021. https://doi.org/10.1016/j.asoc.2021.107439

[15] B. A. S. Emambocus, M. B. Jasser, M. Hamzah, A. Mustapha, A. Amphawan, An enhanced swap sequence-based particle swarm optimization algorithm to solve TSP, *IEEE Access*, Vol. 9, pp. 164820-164836, 2021. https://doi.org/10.1109/ACCESS.2021.3133493

[16] P. Hu, J.-S. Pan, S.-C. Chu, C. Sun, Multi-surrogate assisted binary particle swarm optimization algorithm and its application for feature selection, *Applied soft computing*, Vol. 121, Article No. 108736, May, 2022. https://doi.org/10.1016/j.asoc.2022.108736

[17] J.-S. Pan, Q. Yang, C. S. Shieh, S. C. Chu, Tumbleweed Optimization Algorithm and Its Application in Vehicle Path Planning in Smart City, *Journal of Internet Technology*, Vol. 23, No. 5, pp. 927-945, September, 2022. https://doi.org/10.53106/160792642022092305002

[18] J.-S. Pan, L. G. Zhang, R. B. Wang, V. Snášel, S.-C. Chu, Gannet optimization algorithm: A new metaheuristic algorithm for solving engineering optimization problems, *Mathematics and Computers in Simulation*, Vol. 202, pp. 343-373, December, 2022. https://doi.org/10.1016/j.matcom.2022.06.007

[19] S.-C. Chu, P.-W. Tsai, J.-S. Pan, Cat swarm optimization,

in: Q. Yang, G. Webb (Eds.), *Proceedings of the PRICAI 2006: Trends in Artificial Intelligence: 9th Pacific Rim International Conference on Artificial Intelligence Guilin, China, August 7-11, 2006 Proceedings 9*, Berlin, Heidelberg, 2006, pp. 854-858. https://doi.org/10.1007/978-3-540-36668-3_94

[20] H. Duan, P. Qiao, Pigeon-inspired optimization: a new swarm intelligence optimizer for air robot path planning, *International Journal of Intelligent Computing and Cybernetics*, Vol. 7, No. 1, pp. 24-37, March, 2014. https://doi.org/10.1108/IJICC-02-2014-0005

[21] J.-S. Pan, A.-Q. Tian, V. Snášel, L. Kong, S.-C. Chu, Maximum power point tracking and parameter estimation for multiple-photovoltaic arrays based on enhanced pigeon-inspired optimization with Taguchi method, *Energy*, Vol. 251, Article No. 123863, July, 2022. https://doi.org/10.1016/j.energy.2022.123863

[22] W. Herdianti, A. A. Gunawan, S. Komsiyah, Distribution cost optimization using pigeon inspired optimization method with reverse learning mechanism, *Procedia Computer Science*, Vol. 179, pp. 920-929, 2021. https://doi.org/10.1016/j.procs.2021.01.081

[23] H. Duan, H. Qiu, Advancements in pigeon-inspired optimization and its variants, *Science China Information Sciences*, Vol. 62, No. 7, Article No. 70201, July, 2019. https://doi.org/10.1007/s11432-018-9752-9

[24] Y. Zhong, L. Wang, M. Lin, H. Zhang, Discrete pigeon-inspired optimization algorithm with Metropolis acceptance criterion for large-scale traveling salesman problem, *Swarm and Evolutionary Computation*, Vol. 48, pp. 134-144, August, 2019. https://doi.org/10.1016/j.swevo.2019.04.002

[25] S. Li, H. Chen, M. Wang, A. A. Heidari, S. Mirjalili, Slime mould algorithm: A new method for stochastic optimization, *Future Generation Computer Systems*, Vol. 111, pp. 300-323, October, 2020. https://doi.org/10.1016/j.future.2020.03.055

[26] J.-S. Pan, H.-J. Wang, T.-T. Nguyen, F.-M. Zou, S.-C. Chu, Dynamic reconfiguration of distribution network based on dynamic optimal period division and multi-group flight slime mould algorithm, *Electric Power Systems Research*, Vol. 208, Article No. 107925, July, 2022. https://doi.org/10.1016/j.epsr.2022.107925

[27] S.-C. Chu, X.-W. Xu, S.-Y. Yang, J.-S. Pan, Parallel fish migration optimization with compact technology based on memory principle for wireless sensor networks, *Knowledge-Based Systems*, Vol. 241, Article No. 108124, April, 2022. https://doi.org/10.1016/j.knosys.2022.108124

[28] H. Rezaei, O. Bozorg-Haddad, X. Chu, Grey wolf optimization (GWO) algorithm, in: O. Bozorg-Haddad (Eds.), *Advanced Optimization by Nature-inspired Algorithms*, Springer, Singapore, 2018, pp. 81-91. https://doi.org/10.1007/978-981-10-5221-7_9

[29] J. Shi, J. Shun, Parallel algorithms for butterfly computations, *Symposium on Algorithmic Principles of Computer Systems (APOCS)*, Virtual, pp. 16-30, 2022. https://doi.org/10.1137/1.9781611976021.2

[30] E.-J. Muttio, W.-G. Dettmer, J. Clarke, D. Perić, Z. Ren, L. Fletcher, A supervised parallel optimisation framework for metaheuristic algorithms, *Swarm and Evolutionary Computation*, Vol. 84, Article No. 101445, February, 2024. https://doi.org/10.1016/j.swevo.2023.101445

[31] M. Papadrakakis, N. D. Lagaros, Y. Fragakis, Parallel computational strategies for structural optimization, *International Journal for Numerical Methods in Engineering*, Vol. 58, No. 9, pp. 1347-1380, November, 2003. https://doi.org/10.1002/nme.821

[32] J. P. Cheiney, C. de Maindreville, A Parallel Strategy for Transitive Closure using Double Hash-Based Clustering, *Proceedings of the sixteenth international conference on Very large databases*, Brisbane, Australia, 1990, pp. 347-358.

[33] X. Xia, J. Liu, K. Gao, Y. Li, H. Zeng, Particle Swarm Optimization Algorithm with Reverse-Learning and Local-Learning Behavior, *Chinese Journal of Computers*, Vol. 38, No. 7, pp. 1397-1407, July, 2015. https://doi.org/10.11897/SP.J.1016.2015.01397

[34] H. Wang, H. Sun, C. Li, S. Rahnamayan, J.-S. Pan, Diversity enhanced particle swarm optimization with neighborhood search, *Information Sciences*, Vol. 223, pp. 119-135, February, 2013. https://doi.org/10.1016/j.ins.2012.10.012

[35] H. Wang, H. Li, Y. Liu, C. Li, S. Zeng, Opposition-based particle swarm algorithm with Cauchy mutation, *Proceedings of the 2007 IEEE congress on evolutionary computation*, Singapore, 2007, pp. 4750-4756. https://doi.org/10.1109/CEC.2007.4425095

[36] D. Thierens, Adaptive mutation rate control schemes in genetic algorithms, *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600)*, *IEEE*, Honolulu, HI, USA, 2002, pp. 980-985 . https://doi.org/10.1109/CEC.2002.1007058

[37] J.-S. Pan, Z. Meng, H. Xu, X. Li, QUasi-Affine TRansformation Evolution (QUATRE) algorithm: A new simple and accurate structure for global optimization, in: H. Fujita, M. Ali, A. Selamat, J. Sasaki, M. Kurematsu (Eds.), *Proceedings of the Trends in Applied Knowledge-Based Systems and Data Science: 29th International Conference on Industrial Engineering and Other Applications of Applied Intelligent Systems, IEA/AIE 2016, Morioka, Japan, August 2-4, 2016*, Springer, 2016, pp. 657-667. https://doi.org/10.1007/978-3-319-42007-3_57

[38] Z. Meng, J.-S. Pan, H. Xu, QUasi-Affine TRansformation Evolutionary (QUATRE) algorithm: A cooperative swarm based algorithm for global optimization, *Knowledge-Based Systems*, Vol. 109, pp. 104-121, October, 2016. https://doi.org/10.1016/j.knosys.2016.06.029

[39] K. Panwar, K. Deep, Discrete Grey Wolf Optimizer for symmetric travelling salesman problem, *Applied Soft Computing*, Vol. 105, Article No. 107298, July, 2021. https://doi.org/10.1016/j.asoc.2021.107298

[40] G. Reinelt, TSPLIB—A traveling salesman problem library, *ORSA Journal on Computing*, Vol. 3, No. 4, pp. 376-384, November, 1991. https://doi.org/10.1287/ijoc.3.4.376

[41] Ł. Strąk, R. Skinderowicz, U. Boryczka, A. Nowakowski, A self-adaptive discrete PSO algorithm with heterogeneous parameter values for dynamic TSP, *Entropy*, Vol. 21, No. 8, Article No. 738, July, 2019. https://doi.org/10.3390/e21080738

[42] Y. Cui, J. Zhong, F. Yang, S. Li, P. Li, Multi-subdomain grouping-based particle swarm optimization for the traveling salesman problem, *IEEE Access*, Vol. 8, pp. 227497-227510, December, 2020. https://doi.org/10.1109/ACCESS.2020.3045765

[43] U. Hacizade, I. Kaya, Ga based traveling salesman problem solution and its application to transport routes optimization, *IFAC-PapersOnLine*, Vol. 51, No. 30, pp. 620-625, 2018. https://doi.org/10.1016/j.ifacol.2018.11.224

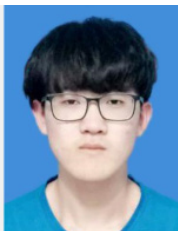
[44] R. T. Bye, M. Gribbestad, R. Chandra, O. L. Osen, A comparison of ga crossover and mutation methods for the traveling salesman problem, in: M. K. Sharma, V. S. Dhaka, T. Perumal, N. Dey, J. M. R. S. Tavares (Eds.), *Proceedings of the Innovations in Computational Intelligence and Computer Vision: Proceedings of ICICV2020*, Springer, 2021, pp. 529-542. https://doi.org/10.1007/978-981-15-6067-5_60

[45] R. R. Asaad, N. L. Abdulnabi, Using local searches algorithms with Ant colony optimization for the solution of TSP problems, *Academic Journal of Nawroz University*, Vol. 7, No. 3, pp. 1-6, August, 2018. https://doi.org/10.25007/ajnu.v7n3a193

[46] N. Rokbani, R. Kumar, A. Abraham, A. M. Alimi, H. V. Long, I. Priyadarshini, L. H. Son, Bi-heuristic ant colony optimization-based approaches for traveling salesman problem, *Soft Computing*, Vol. 25, No. 5, pp. 3775-3794, March, 2021. https://doi.org/10.1007/s00500-020-05406-5

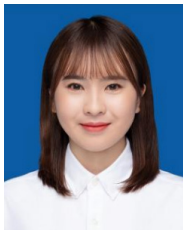
Biographies

Shu-Chuan Chu received the Ph.D. degree in 2004 from the School of Computer Science, Engineering and Mathematics, Flinders University of South Australia. She joined Flinders University in December 2009 after 9 years at the Cheng Shiu University, Taiwan. She is the Research Fellow in the College of Science and Engineering of Flinders University, Australia from December 2009. Currently, she is the Research Fellow with PhD advisor in the College of Computer Science and Engineering of Shandong University of Science and Technology from September 2019. Her research interests are mainly in Swarm Intelligence, Intelligent Computing and Data Mining.



Xiao-Qi Liu received his B.S. degree from North China Institute of Aerospace Engineering in 2022. He is currently pursuing the master degree with the Shandong University of Science and Technology, Qingdao, China. His recent research interests are path optimization and image processing.

Jeng-Shyang Pan received the B.S. degree in electronic engineering from the National Taiwan University of Science and Technology in 1986, the M.S. degree in communication engineering from National Chiao Tung University, Taiwan, in 1988, and the Ph.D. degree in electrical engineering from the University of Edinburgh, U.K., in 1996. He is currently the Director of the Fujian Provincial Key Lab of Big Data Mining and Applications, and an Assistant President with the Fujian University of Technology. He is also the Professor with the Harbin Institute of Technology. He is the IET Fellow, U.K., and has been the Vice Chair of the IEEE Tainan Section. He was offered Thousand Talent Program in China in 2010.



Fei-Fei Liu received her B.S. degree from Shengli College China University Of Petroleum in 2020. She is currently pursuing the master degree with the Shandong University of Scienceand Technology, Qingdao, China. Her recent research interests are swarm intelligence and image processing.

Tien-Szu Pan was born in Taiwan. He received the M.S. and Ph.D. degrees from the University of New Orleans, USA, in 1995 and 1998, respectively. In 2005, he received three champion awards from the International Micro-Robot Competition. In the same year, he was awarded the Best Teaching Faculty at Dayeh University. In 2006, he received another six champion awards from the International Micro-Robot Competition. He is currently a professor at the department of Electronic Engineering at National Kaohsiung University of Science and Technology. His research interests are in the areas of robotics, image processing and mechatronics. He is a member of the IEEE Robotics and Automation Society and SMC.