Real-Coded Quantum Symbiotic Algorithm and Its Application in Cloud Task Scheduling (Post-print)

Authors: Li Kunlun, Guan Liwei

Date: 2018-05-02T00:00:00+00:00

Abstract

To address the issues of slow convergence and susceptibility to local optima in the symbiotic organisms search algorithm, a real-coded quantum symbiotic organisms search (RQSOS) algorithm is proposed by integrating quantum genetic algorithm theory. First, a concept of difference degree is introduced based on triangular fuzzy numbers, and accordingly a triple chromosome is constructed with the components of the independent variable vector and a pair of probability amplitudes as alleles, enabling a single chromosome to carry more information and enhance solution diversity. Then, an exploration learning pattern based on the Archimedean spiral is proposed to strengthen the exploration precision of the solution space. Finally, the symbiotic organisms search algorithm is employed to update the difference degree values, and learning and mutation operations are performed on the population according to these values, promoting rapid evolution of the entire population toward the optimal direction and reducing the probability of falling into local optima. The algorithm is validated through numerical optimization problems and cloud task scheduling problems, and simulation results demonstrate that the RQSOS algorithm achieves significant improvements in both convergence speed and optimization capability, proving it to be a feasible and effective algorithm.

Full Text

Real-Coded Quantum Symbiotic Organisms Search Algorithm and Its Application in Cloud Task Scheduling

Li Kunlun, Guan Liwei
(College of Electronic Information Engineering, Hebei University, Baoding 071000, China)

Abstract

To address the slow convergence and susceptibility to local optima in the symbiotic organisms search algorithm, this paper proposes a real-coded quantum symbiotic organisms search algorithm (RQSOS) by integrating quantum genetic algorithm theory. First, we introduce the concept of difference degree based on triangular fuzzy numbers and construct a triple chromosome with variable vector components and a pair of probability amplitudes as alleles, enabling each chromosome to carry more information and enhance solution diversity. Next, we propose an exploration learning mode based on the Archimedean spiral to strengthen exploration precision in the solution space. Finally, we update the difference degree values using the symbiotic organisms search algorithm and perform learning and mutation operations on the population based on these values, driving the entire population to evolve rapidly toward the optimal direction while reducing the probability of falling into local optima. The algorithm is validated through numerical optimization problems and cloud task scheduling problems. Simulation results demonstrate that RQSOS achieves significant improvements in both convergence speed and optimization capability, proving to be a feasible and effective algorithm.

Keywords: quantum genetic algorithm; symbiotic organisms search; difference degree; numerical optimization; task scheduling

0 Introduction

Quantum genetic algorithm (QGA), first proposed by Narayanan and Moore in 1996, is an intelligent optimization algorithm that combines quantum computing theory with genetic algorithms to enhance algorithmic performance. Subsequently, Han et al. extended it to quantum-inspired evolutionary algorithm (QEA), promoting the integration of quantum computing theory with intelligent algorithms and gradually forming a new research direction—quantum-inspired intelligent algorithms. Representative quantum-inspired intelligent algorithms developed to date include quantum evolutionary programming (QEP), quantum particle swarm optimization (QPSO), and quantum ant colony algorithm (QACO).

Quantum-inspired intelligent algorithms can find better solutions for sorting problems, knapsack problems, path optimization problems, and task scheduling. However, recent research has shown that not all quantum-inspired intelligent algorithms are suitable for solving continuous optimization and multi-parameter optimization problems. For high-precision, high-dimensional computational problems, the binary encoding obtained through quantum bit observation easily leads to chromosome length disaster, resulting in reduced solution precision and decreased algorithmic evolution efficiency. To overcome these drawbacks, real-number encoding-based quantum-inspired intelligent algorithms have gained favor among scholars. These algorithms use quantum bits to encode chromosomes and utilize quantum gates to update the population, over-

coming the high decoding complexity, measurement-induced degradation, and "dimensionality disaster" of traditional quantum-inspired intelligent algorithms by constructing a mapping relationship from quantum bits to real numbers. However, the mapping from quantum bits to real numbers is generally a scaling system where decoding precision cannot be guaranteed, so these algorithms still suffer from low solution precision, slow convergence, and susceptibility to local optima.

To address these issues, this paper combines quantum genetic algorithms with symbiotic organisms search to propose a real-coded quantum symbiotic organisms search algorithm. We validate the algorithm's performance using numerical optimization problems and cloud task scheduling problems. Experimental results show that this approach improves the performance of symbiotic organisms search while successfully avoiding the aforementioned defects.

2 Real-Coded Quantum Symbiotic Organisms Search Algorithm

Real-coded quantum symbiotic organisms search algorithm is an intelligent algorithm that integrates quantum genetic algorithm concepts with the advantages of symbiotic organisms search. To solve the problems of slow early evolution and loss of population diversity in the later stages of symbiotic organisms search, we first improve the quantum bit encoding form by constructing a triple chromosome with correlation characteristics. We then monitor chromosome information to determine population evolution speed and diversity status, and based on monitoring results, timely enhance population evolution speed and supplement population diversity. Finally, we use symbiotic organisms search to update chromosomes and achieve optimal problem solving.

2.1 Chromosome Encoding

This paper proposes an encoding form with correlation characteristics, expressed as:

$$\begin{bmatrix} x_1 & x_2 & \cdots & x_n \\ \cos(\vartheta_1) & \cos(\vartheta_2) & \cdots & \cos(\vartheta_n) \\ \sin(\vartheta_1) & \sin(\vartheta_2) & \cdots & \sin(\vartheta_n) \end{bmatrix}$$

where n is the dimension of the problem to be solved, x_i is the component of the variable vector $\mathbf{x}$, and $\vartheta_i = 0.5 \times \theta_i \times \pi$, with θ being the difference degree value generated by triangular fuzzy numbers. $\cos(\vartheta)$ controls mutation information, while $\sin(\vartheta)$ controls tendency information.

2.1.1 Triangular Fuzzy Numbers Based on set pair theory, the characteristics of triangular fuzzy numbers can be represented by binary connection

numbers $U(x) = x + si$, where x is the mean value and s is the variance, calculated as follows:

$$x = \frac{x_l + x_m + x_n}{3}$$

$$s = \sqrt{\frac{(x_l - x)^2 + (x_m - x)^2 + (x_n - x)^2}{2}}$$

where x_l is the lower bound, x_m is the most likely value, and x_n is the upper bound of the triangular fuzzy number. Here, x and s describe the relative certainty and uncertainty of the triangular fuzzy number $U(x)$, which can be represented in the D-U space as shown in [Figure 1: see original paper].

The argument θ represents the direction of interaction, called the characteristic parameter of $U(x)$, calculated as:

$$\theta = \arctan\left(\frac{s}{x}\right)$$

2.1.2 Population Difference Degree From the analysis in Section 2.1.1, we know that the smaller the θ value ($\theta \in [0, \pi/2]$), the greater the certainty and the smaller the volatility. Therefore, the population difference degree is derived as follows:

Let x_{best} represent the mean of optimal information, which we call the optimal mean; x_{min} is the current dimension minimum, x_{best} represents the current dimension optimum, and x_{max} represents the current dimension maximum. When calculating s', x_m is the current position value, thus calculating the relationship between certainty and volatility. Hence, the difference degree value is defined as:

$$\theta' = \arctan\left(\frac{s'}{x'}\right)$$

where $x' = \frac{x_{min} + x_j + x_{max}}{3}$ and $s' = \sqrt{\frac{(x_{min} - x')^2 + (x_j - x')^2 + (x_{max} - x')^2}{2}}$.

The following example illustrates the relationship between the real-number part of the chromosome and the difference degree:

When diversity is strong in a certain dimension, with 3 as the current optimal value, as shown in the left side of [Figure 2: see original paper], we calculate: $s' = [1.7321, 1.5000, 1.4142, 1.5000, 1.7321]^T$; $\theta = [0.5236, 0.4636, 0.4405, 0.4636, 0.5236]^T$. We can see that the closer the current value is to the optimal mean, the smaller the θ angle value; otherwise, it is larger, but θ remains relatively large.

In the later stage of the population when diversity is small, as shown in the right side of [Figure 2: see original paper], we calculate: $s' = [0.4082, 0.4082, 0.5000, 0.4082, 0.4082]^T$; $\theta = [0.1519, 0.1519, 0.1853, 0.1519, 0.1519]^T$. We can see that θ still satisfies the relationship that the closer to the optimal mean, the smaller the value. In the later algorithm stage, diversity is poor, so the difference degree value is smaller. [Figure 3: see original paper] shows the average difference degree value variation curve of the SOS algorithm optimizing the Rosenbrock function.

2.2 Tendency Learning and Cross Learning

2.2.1 Archimedean Spiral-Based Exploration Learning Mode Literature [16] and [17] have demonstrated the effectiveness of opposition-based learning and rotation learning. However, the setting of rotation angles affects the exploration precision of the algorithm, and this mode still suffers from rough exploration. To further strengthen the algorithm' s ability to approach the optimal solution and its exploration capability, we propose an exploration learning mode based on the Archimedean spiral.

As shown in [Figure 4: see original paper], the rotation mode is divided into two types: (a) Parallel rotation mode, which rotates around the center with the peripheral circle radius, where the intersection point of the radius with the circle and Archimedean line is the parallel learning point; (b) Refined rotation mode, which rotates around the center with the outer circle radius to generate two learning points—one moves from the inner minimum circle of the Archimedean line outward, and the other moves on the outer circle. This creates a pattern where one point learns externally and another learns from the center outward, accelerating the learning rate.

Using point C as the center ($C = (a + b)/2$, where a, b are the solution space bounds) and $(b - a)/2$ as the radius, we construct a peripheral circle and emit an Archimedean spiral within the defined space using equation (13), with the radius varying from 0 to $(b - a)/2$.

The polar coordinate form of the spiral generation function $f(r)$ is:

$$f(r) = a + b\phi$$

where ϕ is the angle and r is the radius.

The learning process involves rotating the current point by a certain angle β. The rotated position becomes the learned point, with its horizontal coordinate:

$$F = \cos(\alpha + \beta) \cdot \left(\frac{b-a}{2}\right) + \frac{a+b}{2}$$

where β is the rotation angle, $\alpha = \arccos\left(\frac{2z-a-b}{b-a}\right)$, and $f(r_i)$ is the rotation position radius. The horizontal coordinate of points on the peripheral circle is:

$$F_1 = \cos(\alpha + \beta) \cdot \frac{b-a}{2} + \frac{a+b}{2}$$

The two learning modes based on the Archimedean line adopt different methods to obtain learning points, which can strengthen the algorithm' s learning precision and improve search efficiency.

2.2.2 Tendency Learning From Section 2.1, we know that when the θ angle value is large ($\sin(\vartheta) > P_t$), population diversity is good, but the comprehensive distance from the current optimal mean is large, representing less optimal information. To strengthen the algorithm' s convergence ability and accelerate its approach to the optimal direction, this section introduces a tendency learning operation that makes individuals approach the optimal individual within a sector space.

The maximum tendency radius is R', as shown in [Figure 5: see original paper], where β_{best} is the angle between the current optimal position on the peripheral circle radius and the x-axis, and β_i is the angle of the current position.

The sector space range and the number of tendency individuals can be adjusted according to actual conditions. The tendency method can be expressed by equations (16)-(18):

$$\beta_{new} = f(\lambda, \beta_i + \Delta\beta)$$

where λ is the acceleration coefficient to determine the number of rotation circles, taking the value $\lambda = d \times 2\pi$, with d being a random integer in $[0, q]$, and q being the specified maximum number of rotation circles.

2.2.3 Cross Learning To further enhance algorithm performance, we propose a cross-learning mode. Specifically, we randomly select M crossover points and use equation (19) to determine whether to learn from the current optimal individual. If learning is effective, the new individual is accepted:

$$x_{i,new}^j = \begin{cases} x_{best}^j, & \text{if } \sin(\vartheta_i) > P_c \\ x_i^j, & \text{otherwise} \end{cases}$$

where $\vartheta = 0.5 \times \theta \times \pi$. This can be interpreted as: when the θ value is large, indicating that the current dimension information is far from the current optimal mean, the algorithm should accept the j-th dimension value of the current optimum with high probability; otherwise, it accepts it with low probability. Deciding whether to learn based on the difference degree value helps improve the success rate of population learning and accelerates population evolution.

2.4 Algorithm Flow

The learning operation guides the entire population to evolve toward the current optimal position. Due to the algorithm' s greedy selection characteristics, population diversity may be lost in later stages, making the algorithm susceptible to local optima. Based on this, we propose a mutation operation.

When the θ value is small, we consider that the population has strong similarity with the current optimum and diversity is lost. At this point, the mutation operation is triggered. Based on the Archimedean exploration mode, we use the following formula to generate mutation information:

$$\Delta\theta_{new} = \Delta\theta_0 \cdot \exp\left(-\mathrm{sgn}(A) \cdot \left|\frac{\nabla f(x)_{max} - \nabla f(x)}{\nabla f(x)_{max} - \nabla f(x)_{min}}\right|\right)$$

where $\nabla f(x)$ is the gradient of the evaluation function at x, $\Delta\theta_0$ is the initial rotation angle (specific value depends on the problem), $-\mathrm{sgn}(A)$ controls the rotation direction, and $A = \sin(\beta_i)\sin(\beta_{best}) + \cos(\beta_i)\cos(\beta_{best})$.

The specific process is as follows:

a) Initialize the real-number population and parameters: maximum rotation circles q, rotation precision a, number of tendency individuals n_1, number of mutation individuals n_2, mutation rate η, etc.

b) Calculate difference degree values through the fuzzy triangular function form and generate the triple chromosome encoding.

c) Use the SOS algorithm to determine individual positions x_i in the solution space and update difference degree values.

d) Learning operations:

- (i) Tendency learning: Generate tendency individuals using equations (16)-(18). If a new individual' s fitness is better than the current individual, replace the current individual with the new one.
- (ii) Cross learning: Generate cross-learning individuals using equation (19). If a new individual' s fitness is greater than the current individual' s, replace the current individual.

e) Mutation operation: Calculate mutation information using equations (20)-(21) and generate mutation individuals to randomly replace n_2 individuals in the population.

f) Check termination conditions. If met, proceed to step g); otherwise, return to step c).

g) Terminate the algorithm and record optimal data.

From the above steps, we can see that RQSOS is an algorithm with a feedback update mechanism, as illustrated in [Figure 6: see original paper]. In the chromosome, the gene values provided by SOS are non-deterministic, with specific values and methods determined by quantum bit probabilities. Therefore, chromosomes in the RQSOS mode can describe the solution space in a superposition state, maintaining solution diversity.

3 Experiments and Analysis

This chapter uses numerical optimization problems and cloud workflow task scheduling problems to compare and analyze the proposed algorithm with other algorithms, verifying its performance.

3.1 Numerical Experiments and Analysis

The experiments use 12 basic test functions listed in to verify algorithm performance, where X_i is the value range, B is the optimal value, D is the function dimension, and Threshold is the solution precision.

The experiments compare the optimization performance and results of QPSO, hybrid particle swarm algorithm (PSO-GA), real-coded quantum differential evolution algorithm (RQDE), and SOS. The parameters for the proposed algorithm are: $q = 12$, $a = 0.25$, $n_1 = 5$, $n_2 = 3$, $\eta = P_t = P_c = \text{rand}$, $\Delta\theta_0 = 0.04\pi$. Each algorithm runs independently 30 times with a maximum of 10,000 iterations. The algorithm terminates when it reaches specified precision or maximum iterations, recording the mean and variance of 30 experiments. The number of successful runs is recorded based on experimental results.

lists the simulation results of the five algorithms, where w/t/l indicates the number of functions where the proposed algorithm wins, ties, or loses compared to the contrast algorithm, M is the mean, and V represents variance.

From , we can see that the proposed algorithm performs best among the five algorithms, obtaining the global optimum for 7 test functions and the best solution for 11 test functions. The basic SOS algorithm only obtains the global optimum for 3 test functions and the best solution for 6 test functions. The RQSOS algorithm outperforms or equals the basic SOS algorithm on all 12 test functions. Except for function f_6 where the optimization effect is slightly worse than QPSO, the proposed algorithm outperforms other algorithms in all cases, proving the effectiveness of our rules.

shows the execution success rates of the five algorithms on different test functions. We can see that RQDE, SOS, and the proposed RQSOS algorithm have better execution success rates than PSO-GA and QPSO, with RQSOS having the highest success rate. This demonstrates the strong exploration ability and good robustness of the SOS algorithm, and further shows that the proposed algorithm maintains the advantages of SOS while overcoming its tendency for premature convergence, achieving better optimization results.

[Figure 7: see original paper] and [Figure 8: see original paper] show the convergence curve comparisons of the five algorithms optimizing functions f_2 and f_8. The results show that different algorithms exhibit different performance on different function optimization problems, while the proposed algorithm consistently demonstrates high convergence ability and strong exploration capability. The reason RQSOS can generate excellent solutions with good convergence is that it adopts learning modes and mutation modes. The learning mode accelerates population evolution toward the optimal solution, while the mutation mode maintains population diversity. Therefore, the proposed algorithm can balance exploration and convergence capabilities, enhancing overall performance.

[Figure 9: see original paper] and [Figure 10: see original paper] compare the population difference degree of SOS and RQSOS algorithms when optimizing functions f_2 and f_8. The results show that when SOS optimizes f_2, population diversity is quickly lost, reducing its ability to explore optimal solutions, resulting in low optimization precision and execution success rate. When optimizing f_8, although diversity can be maintained within a certain range, its approach to the optimum is very slow, making it likely to fail to find the optimal solution within the specified iterations. Clearly, the proposed strategy can control population diversity within a certain range while maintaining high convergence, further proving the algorithm' s effectiveness.

3.2 Application to Cloud Workflow Task Scheduling

Current cloud environment workflow scheduling algorithms process workflows from different perspectives to shorten total completion time and improve system performance. Workflow tasks can be described by a DAG (DAG = {N, S}), where N consists of n node sets representing tasks, and S consists of s edge sets representing constraints between tasks. $S_{ij} \in S$ between n_i and n_j indicates that task n_j must execute after task n_i completes. This paper describes processing cost as task execution time and processor usage fees.

[Figure 11: see original paper] shows a simple DAG graph, where non-negative weights $C_{i,j}$ associated with edge $S_{ij} \in S$ correspond to communication volume between n_i and n_j. If tasks depend on different processors, communication costs between processors must be calculated. Communication costs between tasks on the same processor are considered zero.

The earliest start time $T_{start}(n_i, p_j)$ of task n_i on processor P_j is defined as:

$$T_{start}(n_i, p_j) = \max\{T_{free}(p_j), T_{ready}(n_i)\}$$

where $T_{free}(p_j)$ is the time when processor p_j becomes idle, and $T_{ready}(n_i)$ is the ready time of task n_i. The completion time of task n_i is:

$$FT(n_i) = T_{start}(n_i, p_j) + \frac{W(n_i)}{W(p_j)}$$

where $W(n_i)$ is the computation amount of task n_i, and $W(p_j)$ represents the processing capability of the processor.

If task n_i is a predecessor of task n_j, with p_x and p_y being the processing units for tasks n_i and n_j respectively, the arrival time from n_i to n_j is:

$$AT(n_i, n_j) = FT(n_i) + C(n_i, n_j, p_x, p_y)$$

where $C(n_i, n_j, p_x, p_y)$ is the communication cost. If $i = j$, then $C(n_i, n_j) = 0$.

The priority relationship of task execution on the same processor is determined by:

$$BL(n_i) = W(n_i, p_j) + \max_{n_k \in succ(n_i)} \{C(n_i, n_k) + BL(n_k)\}$$

where M_p and M_c are median processor processing capability and median transmission capability, respectively, $succ(n_i)$ represents the successor set of task n_i, and $BL(n_i)$ is the Bottom-Level of task n_i.

Therefore, the fitness function in this paper can be expressed by formulas (27)-(29):

$$SP = \sum_{j=1}^{m} P_{p_j} \cdot T_{p_j}$$

$$Time = \max_{i=1}^{n} EST(n_i, p_j)$$

$$Fitness = \alpha \cdot Time + \beta \cdot SP \cdot \varepsilon$$

where $\alpha, \beta \in [0, 1]$, $\alpha + \beta = 1$, ε is an adjustment coefficient, T_{p_j} is the total time occupying the j-th processor, and P_{p_j} is the usage price per unit time of the j-th processor. The weights of time and cost can be adjusted according to practical applications.

To verify the effectiveness of the proposed algorithm in handling cloud workflow scheduling problems, we simulate scheduling problems with varying numbers of processors and tasks, and compare with QGA and Improved Quantum Particle Swarm Optimization (IQPSO). In the experiments, the maximum iteration numbers are 400 or 600, the population sizes of QGA and IQPSO are 60, while the proposed algorithm uses a population size of 30, 3 tendency individuals, 5 mutation individuals, circle number $q = 7$, and other parameters are the same as in the previous section.

The experiments use a cloud task scheduling simulation system designed in our laboratory. Based on user-specified parameters, the system randomly generates

10-100 node tasks and 4-10 resource systems. Node computation amounts are controlled within [50, 200], inter-node communication volumes within [5, 30], maximum node in-degree and out-degree of 10, and maximum graph depth of 8. Resource computing capability ranges in [30, 100], resource usage price per unit time ranges in [1, 10], and transmission capability ranges in [10, 30].

[Figure 12: see original paper] and [Figure 13: see original paper] show simulation results when the number of resources is fixed at 8. We can see that IQPSO finds worse completion times than QGA but better processor prices, while the proposed algorithm outperforms both IQPSO and QGA in both time and cost. Scheduling algorithms may produce unbalanced completion times and costs under different constraints, while the proposed algorithm demonstrates better balance.

[Figure 14: see original paper] and [Figure 15: see original paper] show completion times and minimum costs varying with resource numbers when the task number is fixed. We can see that when resources are fewer than 6, the proposed algorithm takes more time than QGA and IQPSO, but when resources exceed 6, it uses less time. For minimum cost, the proposed algorithm outperforms QGA and IQPSO in all cases. Therefore, when the task number is fixed and resource numbers change, the proposed algorithm still demonstrates superior performance.

[Figure 16: see original paper] and [Figure 17: see original paper] show the fitness value changes of the three algorithms when task numbers and resource numbers are fixed. The results show that IQPSO and RQSOS have high early evolution efficiency, but IQPSO performance quickly degrades, while RQSOS maintains high evolution efficiency and outperforms IQPSO. This demonstrates the effectiveness of the proposed learning and mutation modes.

When task numbers increase, task processing becomes complex, and algorithms need certain iterations to find a stable point. However, [Figure 17: see original paper] shows that the proposed algorithm can quickly find better solutions. In later algorithm stages, IQPSO and QGA have the ability to escape local optima but with extremely low efficiency. The proposed algorithm can escape local optima with higher probability in later stages to find better solutions, further demonstrating its superiority.

4 Conclusion

By combining quantum genetic algorithm theory with symbiotic organisms search, this paper proposes a real-coded quantum symbiotic organisms search algorithm. First, we introduce the concept of difference degree and an Archimedean line-based exploration learning mode, enabling the algorithm to search for optimal solutions quickly and meticulously in the solution space. By monitoring difference degree values, we can understand population diversity and distance from the current optimal mean in real time. When population diversity is strong but the comprehensive distance from the current optimal mean is large,

we adopt tendency learning and cross-learning modes to accelerate population evolution toward the optimal direction. When population diversity is lost, to prevent the algorithm from falling into local optima, we introduce mutation operations to strengthen the algorithm' s exploration capability. Numerical experiments and cloud workflow task scheduling experiments demonstrate that the proposed algorithm effectively improves SOS algorithm performance and can obtain better solutions. However, the monitoring information acts on the algorithm in a feedback manner, which increases the original algorithm' s complexity. Therefore, future research should further analyze the model, strengthen the integration between the proposed concepts and the original algorithm, and reduce algorithmic complexity.

References

[1] Narayanan A, Moore M. Quantum-inspired genetic algorithm [C]// Proc of IEEE International Conference on Evolutionary Computation. Piscataway: IEEE Press, 1996: 61-66.

[2] Han K H, Jong H K. Quantum-inspired evolutionary algorithm for a class of combination optimization [J]. IEEE Trans on Evolutionary Computation, 2002, 6 (6): 580-593.

[3] Yang Shuyuan, Jiao Licheng. The quantum evolutionary programming [C]// Proc of the 5th International Conference on Computational Intelligence and Multimedia Applications. 2003: 362-367.

[4] Sun Jun, Xu Wenbo, Feng Bin. A global search strategy of quantum behaved particle swarm optimization [C]// Proc of IEEE Conference on Cybernetics and Intelligent Systems. 2004: 325-331.

[5] Li Panchi, Li Shiyong. Quantum ant colony algorithm for solving continuous problem space optimization problems [J]. Control Theory and Applications, 2008, 25 (2): 237-240.

[6] Luciano R S, Ricardo T, Marley M V. Quantum inspired evolutionary algorithm for ordering problems. Expert Systems with Applications, 2017, 67: 71-83.

[7] Pavithr R, Gursaran S. Quantum inspired social evolution (QSE) algorithm for 0-1 knapsack problem [J]. Swarm and Evolutionary Computation, 2016, 29: 33-46.

[8] Liu Min, Zhang Feng, Ma Yunlong, et al. Evacuation path optimization based on quantum ant colony algorithm [J]. Advanced Engineering Informatics, 2016, 30 (3): 259-267.

[9] Yuan Xiaohui, Wang Pengtao, Yuan Yanbin, et al. A new quantum inspired chaotic artificial bee colony algorithm for optimal power flow problem [J]. Energy Conversion and Management, 2015, 100: 1-9.

[10] Konar D, Bhattacharyya S, Sharma K, et al. An improved quantum-inspired genetic algorithm (HQIGA) for scheduling of real-time multiprocessor system. Applied Soft Computing, 2017, 53: 296-307.

[11] Zhao Shuanfeng, Xu Guanghua, Tao Tangfei, et al. Real-coded chaotic quantum inspired genetic algorithm for training of fuzzy neural networks. Computers and Mathematics with Applications, 2009, 57 (11-12): 2009-2015.

[12] Gao Hui, Xu Guanghui, Zhang Rui, et al. Real-coded quantum evolutionary algorithm [J]. Control and Decision, 2008, 23 (1): 87-90.

[13] Gao Hui, Zhang Rui. Improved real-coded quantum evolutionary algorithm and its application in parameter estimation [J]. Control and Decision, 2011, 26 (3): 418-422.

[14] Cheng Minyuan, Doddy P. Symbiotic organisms search: a new metaheuristic optimization algorithm [J]. Computer and Structures, 2014, 139: 98-112.

[15] Zhao Keqin. Set Pair Analysis and Its Preliminary Application [M]. Hangzhou: Zhejiang Science and Technology Press, 2000.

[16] Rahnamayan S, Wang G G, Ventrescaet M. An intuitive distance based explanation of opposition-based sampling [J]. Applied Soft Computing, 2012, 12 (9): 2828-2839.

[17] Liu Huichao, Wu Zhijian. Differential evolution algorithm based on rotation learning mechanism [J]. Acta Electronica Sinica, 2015, 43 (10): 2040-2046.

Note: Figure translations are in progress. See original paper for figures.