Soft Comput (2017) 21:4387–4398
DOI 10.1007/s00500-016-2071-8

METHODOLOGIES AND APPLICATION

A novel collaborative optimization algorithm in solving complex optimization problems

Wu Deng[1,2,3,4,5] · **Huimin Zhao**[1,2,5] · **Li Zou**[1,3,4] · **Guangyu Li**[1,3] · **Xinhua Yang**[1] · **Daqing Wu**[6,7]

Published online: 18 February 2016

Abstract To overcome the deficiencies of weak local search ability in genetic algorithms (GA) and slow global convergence speed in ant colony optimization (ACO) algorithm in solving complex optimization problems, the chaotic optimization method, multi-population collaborative strategy and adaptive control parameters are introduced into the GA and ACO algorithm to propose a genetic and ant colony adaptive collaborative optimization (MGACACO) algorithm for solving complex optimization problems. The proposed MGACACO algorithm makes use of the exploration capability of GA and stochastic capability of ACO algorithm. In the proposed MGACACO algorithm, the multi-population strategy is used to realize the information exchange and cooperation among the various populations. The chaotic optimization method is used to overcome long search time, avoid falling into the local extremum and improve the search accuracy. The adaptive control parameters is used to make relatively uniform pheromone distribution, effectively solve the contradiction between expanding search and finding optimal solution. The collaborative strategy is used to dynamically balance the global ability and local search ability, and improve the convergence speed. Finally, various scale TSP are selected to verify the effectiveness of the proposed MGACACO algorithm. The experiment results show that the proposed MGACACO algorithm can avoid falling into the local extremum, and takes on better search precision and faster convergence speed.

Keywords Genetic algorithm · Ant colony optimization algorithm · Chaotic optimization method · Multi-strategy · Collaborative optimization · Complex optimization problem

Communicated by V. Loia.

✉ Huimin Zhao
hm_zhao1977@126.com

Wu Deng
dw7689@gmail.com

1 Software Institute, Dalian Jiaotong University, Dalian 116028, China

2 Guangxi Key Laboratory of Hybrid Computation and IC Design Analysis, Guangxi University for Nationalities, Nanning 530006, China

3 The State Key Laboratory of Mechanical Transmissions, Chongqing University, Chongqing 400044, China

4 Traction Power State Key Laboratory of Southwest Jiaotong University, Chengdu 610031, China

5 Key Laboratory of Guangxi High Schools for Complex System & Computational Intelligence, Guangxi University for Nationalities, Nanning 530006, China

6 Department of Computer Science and Technology, University of South China, Hengyang 421001, China

7 Nanjing University of Information Science and Technology, Nanjing 210044, China

1 Introduction

There are a lot of problems in the industry, agriculture, national defense, information and other areas, which can be translated directly into the combinatorial optimization problems (Yanass 2013; Blum et al. 2011). And some of these optimization problems belong to NP-Hard problems. Because the traditional methods are difficult to solve these NP-Hard problems, a lot of intelligent optimization methods were proposed to solve these NP-Hard problems in the past several decades, such as evolution algorithms (Pan et al. 2015; Metaxiotis and Liagkouras 2012) [genetic algorithm (GA), evolutionary programming (EP), differential evolution (DE), etc.], swarm intelligence optimization algorithms (Garnier et al. 2007; Wen et al. 2015) [ant colony optimiza-

tion (ACO), particle swarm optimization (PSO), bee colony optimization (BCO), etc.], and other optimization algorithms (Hamzadayi and Yildiz 2013; Duarte et al. 2011; Robertson et al. 2013) [simulated annealing (SA), tabu, random search method, etc.]. The evolution algorithms use the iterative evolution between populations according to the theoretical basis of the natural evolution to solve complex optimization problems. The swarm intelligence optimization algorithms simulate the behavior of swarm intelligence creatures (bird, bee) in nature to solve complex optimization problems. Due to the different concerns of two types of algorithms, the obtained effects are different in solving complex optimization problems.

With the continuous development of society and economics, the optimization problems are becoming more and more complex, the traditional intelligent optimization methods cannot get the satisfactory solution by using single intelligent optimization method in the finite time. So many researchers proposed hybrid intelligent algorithms based on fusing or combining the different intelligent optimization algorithms for solving complex optimization problems (Corchado and Abraham 2010). Li et al. (2002) proposed a hybrid optimization algorithm based on GA and simulated annealing (SA) to solve a combinatorial optimization problem with constraint. Wei and Zhao (2005) proposed a niche hybrid genetic algorithm (NHGA) based on the niche techniques and Nelder–Mead's simplex method. The proposed method not only makes the exploration capabilities of GA stronger through niche techniques, but also has more powerful exploitation capabilities by using simplex search. Li and Wang (2007) proposed a hybrid quantum-inspired genetic algorithm (HQGA) for the multiobjective flow shop scheduling problem (FSSP). Lee et al. (2008) proposed a novel algorithm of genetic algorithm with ant colony optimization (GA-ACO) for multiple sequence alignment. Tseng and Lin (2009) proposed a hybrid genetic local search algorithm to solve permutation flowshop scheduling problem (PFSP) with each of both criteria. Wen et al. (2011) proposed a heuristic-based hybrid genetic-variable neighborhood search algorithm for the minimization of makespan in the heterogeneous multiprocessor scheduling problem. Xing et al. (2011) proposed a hybrid ACO algorithm (HACOA) to solve instances of the extended capacitated arc routing problem (CARP). This approach is characterized by the exploitation of heuristic information, adaptive parameters, and local optimization techniques. Deng et al. (2012) proposed a novel two-stage hybrid swarm intelligence optimization algorithm called GA-PSO-ACO algorithm that combines the evolution ideas of the genetic algorithms, particle swarm optimization and ant colony optimization for solving traveling salesman problem. Sioud et al. (2012) proposed a hybrid approach based on integrating GA and constraint programming, multi-objective evolutionary algorithms and ant colony optimization for solving a scheduling problem. Sheikhan and Mohammadi (2012) proposed two hybrid models based on combining ACO and GA for feature selection and multi-layer perceptron (MLP). Akpinar et al. (2013) proposed a new hybrid algorithm, which executes ant colony optimization in combination with genetic algorithm (ACO-GA), for type I mixed-model assembly line balancing problem (MMALBP-I) with some particular features of real world problems. Rizk-Allah et al. (2013) proposed a novel hybrid algorithm named ACO-FA, which integrates ACO with firefly algorithm (FA) to solve unconstrained optimization problems. Ghanbari et al. (2013) proposed a new approach, which is called cooperative ant colony optimization-genetic algorithm (COR-ACO-GA), to construct expert systems with the ability to model and simulate fluctuations of energy demand under the influence of related factors. Wang and Duan (2014) proposed a hybrid biogeography-based optimization (HBBO) algorithm based on combining the chaos theory and searching around the optimum strategy with the basic BBO for the job-shop scheduling problem (JSP). Jing (2014) proposed a hybrid genetic algorithm for feature subset selection (HGARSTAR) by embedding a novel local search operation based on rough set theory to fine-tune the search. Mahi et al. (2015) proposed a new hybrid method to optimize parameters that affect performance of the ACO algorithm using PSO. In addition, 3-Opt heuristic method is added to proposed method to improve local solutions. Ali et al. (2015) proposed a hybrid genetic and imperialist competitive algorithm (HGA) to find a near-optimum solution of a nonlinear integer-programming (NIP) with the objective of minimizing the total cost of the supply chain.

Even though these proposed hybrid intelligent algorithms can obtain better optimization performance and solving effects, they still exist falling into the local extremum, low search precision and slow convergence speed. GA is a highly parallel, random and adaptive search algorithm by referring the mechanism of natural selection and natural genetic in the living nature. ACO algorithm is a new optimization algorithm based on the ant colony population to strengthen the local search ability by using the positive feedback of the pheromone. In essence, the GA and ACO algorithm take on the parallelism, robustness and self-organization, so they only rely on the ability of the population to search for the optimal solution. On the basis of analyzing the dynamic characteristics, mechanism, optimization strategies and convergence of the GA and ACO algorithm, the chaotic optimization method, multi-population strategy, adaptive control parameters and collaborative strategy are introduced into the GA and ACO algorithm to realize the information exchange and cooperation among the various populations, avoid falling into the local extremum, dynamically balance the global search ability and local search ability, and improve the search accuracy and convergence speed. So

a genetic and ant colony adaptive collaborative optimization (MGACACO) algorithm is proposed to solve complex optimization problems in this paper. The various scale TSP are selected to verify the effectiveness of the proposed MGACACO algorithm.

The rest of this paper is organized as follows. Section 2 introduces intelligent optimization algorithms, such as genetic algorithm and ant colony optimization algorithm. Section 3 expatiates the chaotic optimization method and multi-strategy, including adaptive control parameters in GA, adaptive adjusting pheromone, dynamic evaporation factor strategy and multi-population collaborative strategy. Section 4 presents a genetic and ant colony adaptive collaborative optimization (MGACACO) algorithm. Section 5 compares our experimental results with the recent algorithms that have been used to solve the TSP. Finally, the conclusions are discussed in Sect. 6.

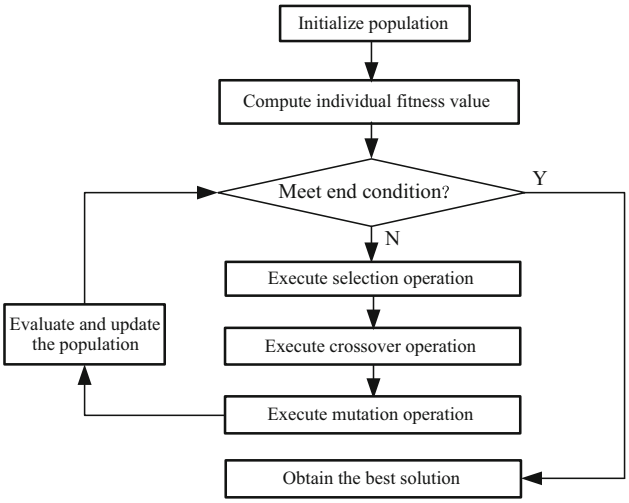

Fig. 1 The flow chart of the GA

2 The intelligent optimization algorithms

2.1 Genetic algorithm

Genetic algorithms (GA) (Holland 1977) is a class of population-based stochastic search techniques that solve optimization problems by imitating processes observed during natural evolution. It is based on the principle of the survival and reproduction of the fitness. The GA continually exploits new and better solutions without any preassumptions, such as continuity and unimodality. The GA is a parallel iterative algorithm with certain learning ability, which repeats evaluation, selection, crossover and mutation operators until the stopping criteria or maximum number of iterations are reached. It has been widely applied in solving many complex optimization problems, such as function optimization, multi-objective optimization, traveling salesmen problem and so on. The GA shows its merits in solving optimization problems; especially it is propitious to the problems of multiple optimum solutions. A real-coded GA is a genetic algorithm representation that uses a vector of floating-point number instead of 0 and 1 for implementing chromosome encoding. With some modifications of the genetic operators, the real-coded GA has better performance than the binary-coded GA in solving traveling salesman problem. The crossover operator of the real-coded GA is performed by borrowing concept of convex combination. Assuming that the GA is employed to search for the largest fitness value with the given fitness function, the flow of the GA is shown in Fig. 1.

2.2 Ant colony optimization (ACO) algorithm

Ant colony optimization (ACO) algorithm was proposed by Dorigo and Gambardella (1997). It is a metaheuristic inspired by the behavior of real ants in search for the shortest path to food sources. Ants tend to choose the paths marked by the strongest pheromone concentration. The ACO algorithm is an essential system based on agents that simulates the natural behavior of ants, including the mechanisms of cooperation and adaptation. It simulates the techniques employed by real ants to rapidly establish the shortest route from a food source to their nest and vice versa without the use of visual information. The ACO algorithm consists of a number of cycles (iterations) of solution. In each iteration, a number of ants construct complete solutions by using heuristic information and the collected experiences of previous population of ants. These collected experiences are represented by using the pheromone trail, which is deposited on the constituent elements of a solution. The pheromone can be deposited on the components and/or the connections used in a solution depending on the solving problem. The flow of ACO algorithm is illustrated in Fig. 2.

3 Chaotic optimization method and multi-strategy

3.1 Chaotic optimization method

Chaos is an interesting nonlinear dynamics and random behaviour in the system (Fister et al. 2015). It has characteristics of the randomicity, ergodicity, sensibility, and so on. The basic idea of chaos is to linearly map chaotic variable into the value range of optimization variable. Chaotic optimization method is one of the effective methods for solving function optimization problem (Okamoto and Hirata 2013). Due to the ergodicity of phase space and intrinsic randomness in the chaos, the chaotic variables can avoid falling into the local minimum and overcome the shortcomings of traditional optimization algorithm. In addition, the chaotic optimization

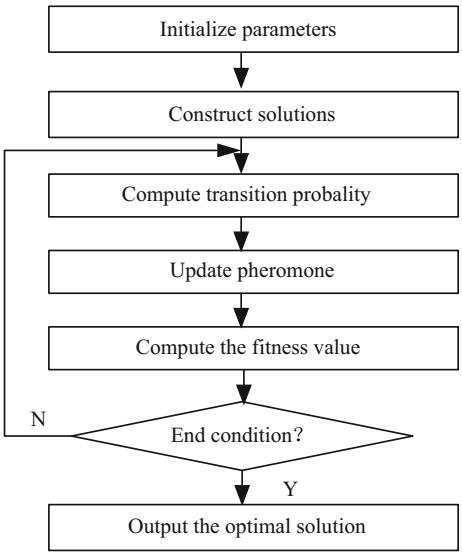

Fig. 2 The flow chart of the ACO algorithm

method has simple structure and easy implementation and so on. Logistic is a typical chaotic system, its mapping has characteristic of the definite expression, so the chaos system does not contain any random factors. In this paper, a one-dimensional Logistic mapping method is used to generate sequences. The mathematical expression of one-dimensional Logistic map method is given (Kromer et al. 2014):

$$x(t+1) = \mu x(t)(1 - x(t)) \quad t = 1, 2, 3, \ldots, \mu \tag{1}$$

where the control variable $\mu \in [0, 4]$ is the parameter of Logistic. When control variable $\mu = 4$, Logistic mapping is full mapped on [0,1], and the Logistic mapping is complete chaotic state. $z_1, z_2, z_3, \ldots$ is obtained a determined time series by using the arbitrary initial value z_0. The characteristic of chaotic motion is used to optimize and search. The basic idea is to generate a set of chaotic variables with the same number of optimization variables, then the chaos is introduced into the optimization variables by using the chaotic carrier mode to obtain the chaotic state.

3.2 Adaptive control parameter in GA

Be aimed at balancing the search range and search ability in GA, a lot of improved GAs are proposed in recent decades, such as adaptive genetic algorithm, immune genetic algorithm, hybrid genetic algorithm and so on. But the adaptive genetic algorithm cannot promptly reflect the individual's premature convergence. If the individuals with the lower fitness value and local optimum are used to form one population, then the adaptive genetic algorithm will mistakenly believe that the population do not discover the premature convergence. It cannot escape from the local optimal solution and reduce the search optimization performance. If the fitness value of individual is greater than the average fitness value of population ($\overline{f(t)}$), the fitness values of these the individuals will redo the mean to obtain the average value ($\overline{\overline{f(t)}}$). Define $\phi = f_{\mathrm{GA}}^{\max}(t) - \overline{\overline{f(t)}}$, which represents the premature convergence extent of population. Then the crossover probability $P_{\mathrm{c}}(t)$ and mutation probability $P_{\mathrm{m}}(t)$ are adaptively controlled, the expressions are given:

$$P_{\mathrm{c}}(t) = \begin{cases} \frac{k_1(f_{\mathrm{GA}}^{\max}(t) - f'_{\mathrm{GA}}(t))}{f_{\mathrm{GA}}^{\max}(t) - \overline{\overline{f(t)}}}, & f'_{\mathrm{GA}}(t) \geq \overline{\overline{f(t)}} \\ k_3, & f'_{\mathrm{GA}}(t) \leq \overline{\overline{f(t)}} \end{cases} \tag{2}$$

$$P_{\mathrm{m}}(t) = \begin{cases} \frac{k_2(f_{\mathrm{GA}}^{\max}(t) - f_{\mathrm{GA}}(t))}{f_{\mathrm{GA}}^{\max}(t) - \overline{\overline{f(t)}}}, & f_{\mathrm{GA}}(t) \geq \overline{\overline{f(t)}} \\ k_4, & f_{\mathrm{GA}}(t) \leq \overline{\overline{f(t)}} \end{cases} \tag{3}$$

where the coefficients of k_1, k_2, k_3 and k_4 are less than or equal to 1. The $f'_{\mathrm{GA}}(t)$ is the larger fitness function value in two crossover individuals. The $f_{\mathrm{GA}}(t)$ is used function value of mutation individual.

3.3 Adaptive and dynamic control parameters in ACO

3.3.1 The search optimization strategy

In the route, the kth ant starts from city r, the next city s is selected among the unvisited cities memorized in J_r^k according to the following expression (Otero et al. 2013):

$$j = \begin{cases} \arg\max\{[\tau_{ij}(t)][\eta_{ij}(t)]^{\beta}\}, & q(t) \leq q(\lambda(t)) = \lambda(t)/N \\ S, & \text{otherwise} \end{cases} \tag{4}$$

where $\lambda(t) \in [2, N]$ presents ANB of the algorithm in the tth iteration, N is the number of nodes. $q(\lambda(t)) \in [2/N, 1]$, $q(t)$ is uniformly distributed random number on [0,1].

The next city s is selected according to the following expression:

$$p_k(i, j) = \begin{cases} \frac{[\tau(i,j)]^{\alpha}[\eta(i,j)]^{\beta}}{\sum_{u \in J_k(i)}[\tau(i,j)]^{\alpha}[\eta(i,u)]^{\beta}} & j \in J_k(i) \\ 0 & \text{other} \end{cases} \tag{5}$$

where $p_k(i, j)$ is the transition probability (from city i to city j for the kth ant), $\tau(i, j)$ is the intensity of pheromone between city i and city j, $\eta(i, j)$ is the length of path between cities from city i to city j, $J_k(i)$ is a set of unvisited cities of the kth ant, the parameter α and β are the control parameters for determining the weight of trail intensity and the length of path, q is a random uniform probability in [0, 1], and q_0 is a parameter between 0 and 1 .

$$\alpha = \begin{cases} \alpha \left[1 + \text{rand}(-|\sin(t)|, |\sin(t)|) * e^{\frac{-(t-N_c\max/2)^2}{2}} \right] & \alpha \in [\alpha_{\min}, \alpha_{\max}] \\ \alpha_c & \alpha < \alpha_{\min} \text{ or } \alpha > \alpha_{\max} \end{cases} \tag{6}$$

$$\beta = \begin{cases} \beta * |\sin(t)| + 1.01 & \beta \in [\beta_{\min}, \beta_{\max}] \\ \beta_c & \beta < \beta_{\min} \text{ or } \beta > \beta_{\max} \end{cases} \tag{7}$$

where α_c is the initial value of α, $\alpha_{\min}$ and $\alpha_{\max}$ indicate respectively the minimum value and maximum value of the α. β_c is the initial value of β, $\beta_{\min}$ and $\beta_{\max}$ indicate respectively the minimum value and maximum value of the β. rand() is a random function.

3.3.2 Adaptive adjusting pheromone

In the traditional ACO algorithm, a fixed amount of pheromone is used to update the pheromone on the path. This updating strategy ignores the distribution characteristics of solution and is prone to stagnation and slow convergence speed. So adaptive adjusting pheromone strategy is used to make relatively uniform distribution of pheromone and effectively solve the contradiction between expanding search and finding optimal solution to search for the local optimal solution in this paper.

The real variable function $Q(t)$ is used to replace the constant of pheromone intensity Q in the adjusting pheromone $\Delta\tau_{ij}^k = Q/L_K$ to balance the exploration and exploitation between the random search of ant and the evocation function of information under the pheromone evaporation or increasing in the random search process. The adaptive adjusting pheromone expression is given:

$$\Delta\tau_{ij}^k(t) = Q(t)/L_K \tag{8}$$

$$Q(t) = \begin{cases} Q_1 & t \le T_1 \\ Q_2 & T_1 < t \le T_2 \\ Q_3 & T_2 < t \le T_3. \end{cases} \tag{9}$$

In the adaptive adjusting pheromone strategy, if the obtained optimal solution does not change in a period of time, it shows that the search falls into an extreme point. Then the enforcement mechanism is adopted to decrease the amount of increasing information to escape from local minimal value. The amount of pheromone on the optimal path and worst path are reduced to avoid falling into local optimal solution in the initial stage of search process. Then the positive feedback of ACO need be appropriately restrained by adding a small amount of negative feedback pheromone in the search process, the goal is to reduce the difference of pheromone and expand the scope of the search.

3.3.3 Adaptive control parameters $q(t)$

In the traditional ACO algorithm, the value of stochastic selection threshold $q(t)$ is a random number on [0, 1]. Because the diversity of the solution is increasing, it will reduce to fall into the local optimum in a certain extent. But it is difficult to control the stochastic selection threshold $q(t)$. So the uniformly distributed random number on [0, 1] is used to control the stochastic selection threshold $q(t)$. The expression is given:

$$q(t+1) = \begin{cases} q(t)/\mu, & q(t) \in (0, \mu) \\ (1 - q(t))/(1 - \mu), & q(t) \in (\mu, 1) \end{cases} \tag{10}$$

where $\mu \in (0, 1)$.

3.3.4 Dynamic evaporation factor strategy

The evaporation factor of pheromone ρ in the basic ACO algorithm is a constant. The ρ value directly relates to the global search ability and convergence speed. For large-scale problems, because there exists the evaporation factor of pheromone, the pheromone on the unvisited path will be reduced to close to 0. This will reduce the global searching ability of ACO algorithm. If the pheromone is too large, the selection probability of visited path will be large, this also will affect the global search ability of ACO algorithm. Therefore, how to set the value of pheromone has become the key to control the pheromone releasing and evaporating. The concept of dynamic evaporation rate is used in this paper. The idea is to set a larger value for dynamic evaporation rate ρ at the beginning of ACO algorithm to enhance the global search ability (Jovanovic and Tuba 2011). But with the operation of ACO algorithm, the evaporation rate ρ will continue to decay, so that the ACO algorithm can quickly converge to the optimal solution. The dynamic evaporation factor strategy in ACO algorithm not only increases the global search capability, but also accelerates the convergence in a certain extent.

To better explore decay model of evaporation rate, there are three different decay models of curve decay model, line decay model and scale decay model. The curve decay model is selected according to implementing a set of experiments. The expression of curve decay model is described as follows:

$$\rho(t) = \frac{T \times (\tau_{\max} - \tau_{\min}) \times t}{T - 1} + \frac{T \times \tau_{\min} - \tau_{\max}}{T - 1} \quad (11)$$

$$\tau(t) = (1 - \rho(t)) \times \tau(t) + \sum_{k=1}^{m} \Delta\tau_{ij}^{k}(t) \quad (12)$$

where $\tau_{\max}$ and $\tau_{\min}$ respectively are the upper and lower of pheromone. t and T respectively refer to the current iteration and the maximum iteration.

3.4 Multi-population collaborative strategy

Multi-population collaborative strategy is to divide the population into several sub-populations for collaborative evolution. The multi-population is used to realize the information exchange and cooperation among the various populations. The collaborative evolution mode has two kinds of common models: the island model and neighborhood model. The two models directly divide the individuals into several sub-populations. The collaborative strategy is used to dynamically balance the global ability and local search ability, and improve the convergence speed. Each sub-population represents a subspace in the solution space, and is optimized and updated according to the respective search strategy. The searched better individual will be migrated among the different sub-populations, and is regarded as a shared information to guide the evolution to effectively improve the searching efficiency. So the multi-population collaborative strategy takes on strong global and local search ability, ensures the strong spatial exploration and exploitation ability in the search process of particles, and accelerates the search speed of population, improves the convergence precision, accuracy and efficiency in the optimization process.

4 A genetic and ant colony adaptive collaborative optimization (MGACACO) algorithm

4.1 The idea of the proposed MGACACO algorithm

GA is a parallel, random and adaptive search algorithm. It applies the mechanism of natural selection and natural genetic in the process of biological evolution to solve the optimization problems. And it has the advantages of self-adaptation, parallelism and strong global convergence ability. But it exists the weak local search ability. The ACO algorithm is a new optimization algorithm based on the ant colony. It has been proved to be a very promising and effective method in solving complex optimization problems, especially for discrete optimization problems. It has the advantages of distributing, self-organizing, positive feedback. But it inevitably has some defects, such as the slow speed of initial pheromone generation, poor global convergence and so on. And the unbalance between the global search and the local search in ACO algorithm could cause the premature phenomenon and the stagnation phenomenon. In essence, the GA and ACO algorithm take on the parallelism, robustness and self-organization, they only rely on the ability of population to search for the optimal solution. Currently, several algorithms are integrated according to the certain rules or optimization idea to construct hybrid optimization algorithms, which can effectively overcome their weaknesses and fully use their advantages for improving the optimization performances. So the chaotic optimization method, multi-population collaborative strategy and adaptive control parameters are introduced into the GA and ACO algorithm to propose a genetic and ant colony adaptive collaborative optimization (MGACACO) algorithm for solving complex optimization problems in this paper. In the proposed MGACACO algorithm, the population is divided into several subpopulations, then the GA and ACO algorithms are guided each other by updating the global optimization solution. In the GA, a population is regarded as ant colony, and then one individual in the population is regarded as an ant in the ant colony. The obtained optimal solution by using the crossover and mutation operations is used to describe the selected action of the path of the ant. In the ACO algorithm, each ant selects one path according to the heuristic function and the consistence of pheromone. The ant colony uses the left pheromone to achieve the collaborative optimization purpose. Each ant is regarded as an individual in the GA. The ant colony is regarded as the population in each generation in the GA. The ACO algorithm emphasizes the search process of the individual. The collaborative optimization is obtained by using positive feedback operation of left pheromone. The GA is used to dynamically adjust the initial pheromone distribution of ACO algorithm to better control the exploitation ability and exploration ability of each subpopulation. Then the adaptive adjusting pheromone is introduced into each subpopulation to improve the exploration ability and avoid prematurely converging to the local optimal solution. The information exchange is the exchange operation of the pheromone matrix among subpopulations. And the collaborative strategy is used to dynamically balance the global search ability and local search ability, comprehensively evaluates the state of ACO algorithm and GA. The proposed MGACACO algorithm can obtain the global search ability of GA and the local search ability and convergence speed of ACO algorithm by the multiple cycles. The corresponding updating method is used to improve the optimization performance of the proposed MGACACO algorithm.

4.2 The steps of the propose MGACACO algorithm

First, the running states of the GA and ACO algorithm are comprehensively evaluated, then the proposed MGA-

CACO algorithm adaptive selects the GA or ACO algorithm. According to the previously defined convergence speed and diversity function, we define a function $F(g)$, which describes the running state of the gth generation of the MGACACO proposed algorithm.

$$F(g) = \begin{cases} d(\vec{x}) < C_1 & \text{GA} \\ |\text{Conv}(g)| \bullet \text{Diff}_{g,g-1} < C_2 & \text{ACO} \end{cases} \tag{13}$$

where g denotes the number of iteration, C_1 and C_2 are constant. The value of the $F(g)$ is Boolean value. If the obtained result is false, the current algorithm continues to implement. Otherwise another algorithm is selected to implement. The description of the proposed MGACACO algorithm is given:

Step 1 Initialize the MGACACO algorithm.

For solving complex optimization problem, the size of initial population (M_g) is determined according to the constraints of the dimension, search point and rate and so on. The relative parameters of the MGACACO algorithm are set: initial crossover probability (P_c), initial mutation probability (P_c), the max number of the iteration ($T_{\max}$), control parameters α_c, $\alpha_{\min}$ and $\alpha_{\max}$, β_c and $\beta_{\min}$ and $\beta_{\max}$, pheromone trial evaporation rate ρ_0, stochastic selection threshold (q_0), the upper and lower of pheromone $\tau_{\max}$ and $\tau_{\min}$, the initial iteration $t = 0$ and so on. Real number coding method is used in this paper.

Step 2 Calculate, compare and update the fitness value.

Calculate the fitness value of each individual, the tournament selection method is used to select the initial population to find a number of optimal solutions. The average fitness value is calculated according to the followed expression.

$$\overline{F}_G(t) = \frac{1}{M_G} \sum_{i=1}^{M_G} F_G^i(t) \tag{14}$$

$$\overline{F}_A(t) = \frac{1}{M_A} \sum_{i=1}^{M_A} F_A^i(t) \tag{15}$$

where $F_G^i(t)$ and $F_A^i(t)$ are the fitness function values of the ith solution.

Step 3 Generate the chaotic variables.

The chaotic signal generator is used to generate the chaotic variables according to the expression (1), which are used to initialize the pheromone distribution to guarantee the balance the global optimization and the local optimization.

Step 4 Obtain the crossover probability and mutation probability.

The parameters of crossover probability $P_c(t)$ and mutation probability $P_m(t)$ in GA are adaptively controlled according to the expression (2) and (3) in this iteration.

Step 5 Execute mutation operation.

Each parent individual can generate a sub-individual according to the following expression.

$$x_i' = (x_i + \lfloor n! | N(0,1) \rfloor) \bmod n! \tag{16}$$

Step 6 Obtain several optimal solutions.

The obtained fitness values are compared with other optimal solutions to obtain the global optimal value.

Step 7 The obtained optimal solutions are used to initialize the pheromone on each path to obtain the value of pheromone of each path. At the same time, the ants (M_A) are randomly placed in the N nodes.

Step 8 The chaotic signal generator is used to generate the uniformly distributed random number ($q(t)$) on [0, 1]. Each ant of ACO algorithm selects the next visiting city according to the expression (5).

Step 9 The pheromone is updated according to the expression (12).

Step 10 Calculate the completed path length.

The visited path length of each ant and the fitness value of solution are calculated. In this iteration, the optimal solution $S_b(t)$ is saved. If the value of $S_b(t)$ is better than S_b, then the value of S_b is replaced by the value of $S_b(t)$. Otherwise return to Step.6.

Step 11 Obtain parameters of ACO algorithm.

The parameters of obtained pheromone trial evaporation rate $\rho(t)$, stochastic selection threshold $q(t)$, and control parameters α and β in ACO algorithm adaptively controlled according to the expression (6)–(9) and (11).

Step 12 Calculate the average value $\overline{F}_A(t)$.

The average value $\overline{F}_A(t)$ of the obtained solutions is calculated and recorded according to the expression (15).

Step 13 The fitness function value and the average value of the proposed MGACACO algorithm are calculated according to the following expression. The fitness value of obtained individual is greater than the value of $\overline{F}(t)$,then the obtained individual replaces the individual with the lower fitness value in the initial population.

$$F(t) = \lambda \overline{F}_G(t) + \mu \overline{F}_A(t) \tag{17}$$

$$\lambda = \frac{F_G^{\max}(t) - F_G^{\min}(t)}{(F_G^{\max}(t) + F_A^{\max}(t)) - (F_G^{\min}(t) + F_A^{\min}(t))} \tag{18}$$

$$\mu = \frac{F_A^{\max}(t) - F_A^{\min}(t)}{(F_G^{\max}(t) + F_A^{\max}(t)) - (F_G^{\min}(t) + F_A^{\min}(t))} \tag{19}$$

Step 14 The value of fitness function $F(t+1)$ of all solutions are calculated according to the expression (17), the optimal

solution in this iteration is recorded and is compared with the history optimal solution. If the obtained optimal solution is better than the history optimal solution, the obtained optimal solution will replace the history optimal solution. Otherwise return Step 12.

Step 15 The average value of the fitness function $F(t+1)$ is calculated. And determine whether the termination condition of the algorithm is met. If the termination condition is met, the history optimal solution is outputted. Otherwise $t = t+1$, and return Step 5.

Step 16 Output the results.

4.3 Analyze the convergence and diversity of the proposed MGACACO algorithm

In the running GA or ACO algorithm, there is easy to appear slow convergence speed and stagnation phenomenon. The reason is that the diversity of population tends to coincide. So an evaluation function is defined to respectively analyze the convergence speed and diversity of the proposed MGACACO algorithm.

In the diversity of population of GA, the $d(\vec{x})$ is used to describe the diversity:

$$d_j(\vec{x}) = \left| \bigcup_{i=1}^{n} x_{ij} \right| \tag{20}$$

$$d_j(\vec{x}) = \frac{1}{l} \sum_{j=1}^{l} d_j(\vec{x}) \tag{21}$$

where $\vec{x} = x_1, x_2, \ldots x_n)$ is population, $|\bigcup_{i=1}^{n} x_{ij}|$ denotes different individuals of the population at the position of the x_j.

In the diversity of the population of ACO algorithm, the $P(g)$ is used to describe the diversity:

$$P(g) = \bigcup_{k=1}^{n} P_k(g) \tag{22}$$

$$\text{Diff}_{ij} = \frac{C}{|P(i) - P(j)| + |P(j) - P(i)} \tag{23}$$

where Diff_{ij} denotes the difference of ACO algorithm between the search path result of the kth generation and the search path result of the jth generation.

So the diversity of the proposed MGACACO algorithm is represented by following expression:

$$\text{DIV} = d_j(\vec{x}) + P(g) = \left| \bigcup_{i=1}^{n} x_{ij} \right| + \bigcup_{k=1}^{n} P_k(g). \tag{24}$$

The convergence speed of the proposed MGACACO algorithm is represented by following expression:

$$D(g) = \frac{1}{n} \sum_{k=1}^{\infty} d_k(g) \tag{25}$$

$$\text{Conv}(g) = \frac{1}{n} \sum_{k=1}^{\infty} d_k(g) - \frac{\sum_{k=1}^{g-1} D(k)}{g-1}. \tag{26}$$

5 Experimental results and analysis of the MGACACO algorithm

5.1 Operation environment and parameter configuration

To test the performance of the proposed MGACACO algorithm, several instances from TSPLIB (http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/) are selected in this psper. According to the characteristics of TSPLIB, the distance among any two cities is calculated by the Euclidian distance. At the same time, the GA, ACO and PGACS (Jovanovic and Tuba 2011; Chen and Chien 2011) (parallelized genetic ant colony system) algorithms are selected to compare with the proposed MGACACO algorithm. The experiment environment is described: the Pentium IV, 2.40 GHz, 2.0 GB RAM, Windows XP and Matlab 2012b. The initial parameters of the GA, ACO, PGACS and MGACACO algorithms are selected after thorough testing. In the simulation experiments, the alternative values were tested and modified for some functions to obtain the most reasonable initial values of these parameters. These selected values of the parameters take on the optimal solution and the most reasonable running time of these algorithms to efficiently complete the problem solving. So the selected values of these parameters are shown in Table 1

5.2 Experimental results

The GA, ACO, PGACS and MGACACO algorithms are performed on a set of 28 Euclidean sample problems with sizes ranging from 29 to 18,512 nodes. Each instance is described by its TSPLIB name and size, e.g. in Table 2 the instance named bayg29 has size equal to 29 nodes. For each instance, a batch of 30 independent executions is performed. The results of the simulated experiments are shown in Table 2. In Table 2, Best represents the found length of best solution, avg. represents the found average length of all solutions.

As it can be observed from Table 2, for the 28 TSP instances with the GA, ACO, PGACS and MGACACO algorithms, the best values and average values of the MGACACO algorithm are better than the best values and average values of the three other optimization algorithms in the experiment. For TSP instances eil51 and rad100, the proposed MGACACO algorithm can find the best known solutions 426

Table 1 Parameters of the GA, ACO, PGACS and MGACACO algorithms

Parameters	GA	ACO	PGACS	MGACACO
Population size (M_g)	200	200	200	200
Ants (M_A)	N/A	20	20	20
Iteration time ($T_{\max}$)	1000	1000	1000	1000
Crossover probability (P_c)	0.90	N/A	0.90	0.90
Chaotic search state (μ)	N/A	N/A	N/A	3
Mutation probability (P_m)	0.01	N/A	0.01	0.01
Pheromone factor (α)	N/A	1.0	1.0	1.0
Heuristic factor (β)	N/A	2.0	2.0	2.0
Evaporation coefficient [$\rho(t_0)$]	N/A	0.05	0.05	0.05
Stochastic selection threshold (q_0)	N/A	0.1	0.1	0.1
Pheromone amount (Q)	N/A	100	100	100

Table 2 The results of the simulated experiments

Instances	Opt.	GA		ACO		PGACS		MGACACO	
		Best	Avg.	Best	Avg.	Best	Avg.	Best	Avg.
bayg29	1272	1375.68	1397.49	1302.47	1325.64	1286.32	1298.56	1275.67	1284.36
eil51	426	445.46	467.43	431.02	437.79	429.45	432.29	**426.00**	**427.21**
berlin52	7542	76,120.45	7673.37	7584.36	7602.83	7579.24	7596.16	7544.37	7544.37
eil76	538	573.461	569.86	552.462	569.865	546.604	563.509	**539.153**	**540.302**
rat99	1211	1310.35	1335.32	1262.53	1294.32	1248.94	1277.79	1216.56	1223.86
rad100	7910	8089.25	8136.94	7986.36	8003.61	7964.38	7995.92	**7910.00**	**7934.69**
eil101	629	698.589	714.761	681.562	693.472	666.749	680.683	**630.01**	**634.68**
lin105	14,379	14,513.6	14,554.5	14,464.3	14,483.8	14,411.2	144,513	14,381.8	14,394.1
ch130	6110	6198.39	6199.37	6148.32	6171.66	6128.19	6145.94	6113.26	6123.92
pr136	96,772	97,147.2	67,331.6	96,989.5	67,279.3	96,943.6	67,202.3	96,945.1	67,142.6
kroA150	26,524	26,852.3	26,951.4	26,614.7	26,659.6	26,578.4	26,595.1	26,524.7	26,537.4
u159	42,080	42,810.7	42,967.2	42,489.2	42,503.9	42,430.6	42,490.4	42,381.6	42,413.5
kroA200	29,368	29,856.1	29,985.6	29,505.3	29,521.8	29,470.7	29,492.8	29,380.2	29,434.7
pr226	80,369	80,913.6	89,003.1	80,689.2	80,821.5	80,612.6	80,785.3	80,532.1	80,692.3
pr264	49,135	49,793.5	49,963.7	49,414.5	49,498.2	49,325.1	49,430.4	49,138.0	49,165.1
pr299	48,191	48,741.5	48,891.2	48,395.4	48,487.9	48,313.7	48,405.9	48,246.1	48,313.5
lin318	42,029	42,803.3	42,971.7	42,514.3	42,586.3	42,451.7	42,490.2	42,284.9	42,368.3
rd400	15,281	15,754.3	15,910.7	15,452.6	15,521.2	15,386.0	15,432.5	15,332.6	15,394.7
pcb442	50,778	51,547.9	51,710.3	51,216.7	51,297.9	51,101.5	51,230.9	50,982.5	51,196.3
rat575	6773	6998.25	7094.58	6910.52	6999.95	6892.42	6970.56	**6859.85**	6901.25
u724	41,910	42,852.6	42,964.8	42,412.7	42,513.8	42,291.4	42,398.3	42,267.7	42,378.5
rat783	8806	9098.67	91,689.3	8996.35	9012.55	8985.47	8995.21	8940.37	8976.92
pr1002	259,042	265,681	266,047	264,130	265,671	263,652	265,029	263,276	264,195
fl140	20,127	20,985.3	21,048.6	20,812.2	20,884.7	20,741.7	20,780.6	20,683.8	20,776.2
d1655	62,128	64,106.5	64,506.2	63,941.1	64,346.4	63,701.6	64,170.4	**63,615.6**	64,133.7
pcb3038	137,694	143,196	144,867	141,103	141,958	141,129	141,584	140,589	141,047
Rl5934	556,045	577,841	578,954	574,562	576,120	573,448	575,886	572,348	575,371
d18512	645,238	6,730,673	67,668	669,741	674,925	668,474	671,200	667,649	669,265

Better values found using the proposed algorithm are in bold

Table 3 The comparison results of the MGACACO algorithm with three other algorithms

Instances	Optimal value	Pasti's method (2006)		Marinakis' method (2010)		Escario' method (2015)		MGACACO	
		Best	Avg.	Best	Avg.	Best	Avg.	Best	Avg.
eil51	426	433.00	442.72	430.16	437.79	427.12	429.68	**426.00**	**427.21**
eil76	538	552.00	563.20	544.672	551.241	540.538	545.690	**539.153**	**540.302**
rad100	7910	7937	8342.80	7935.46	7968.25	7912.65	7935.46	**7910.00**	**7934.69**
eil101	629	640.05	665.93	635.68	650.43	631.76	636.04	**630.01**	**634.68**
lin105	14,379	14,379	16,031.3	14,395.1	14,414.5	**14,380.1**	14,393.9	14,381.8	14,394.1
ch130	6110	6203	6332.43	6179.03	6199.35	**6112.46**	**6120.68**	6113.26	6123.92
ch1150	6528	6631	6742.46	6581.47	6597.15	6531.76	6561.32	**6528.00**	6539.86
kroB150	26,130	26,342	28,404.3	26,184.1	26,210.3	26,146.1	26,178.3	**26,139.4**	26,175.3
kroA200	29,368	30,144	33,228.3	29,404.2	29,476.3	**29,376.2**	29,402.9	29,380.2	29,434.7
kroB200	29,437	29,703	33,838.1	29,616.3	29,690.6	29,531.3	29,574.7	**29,502.6**	29,512.4
lin318	42,029	43,154	45,832.8	**42,275.6**	42,331.8	42,280.4	42,374.1	42,284.9	42,368.3
rat575	6773	7090	8301.8	6902.94	6947.17	6867.02	6921.37	**6859.85**	6901.25
rat783	8806	9316	10,721.6	9104.65	9126.01	**8806.00**	**8806.72**	8940.37	8976.92
fl140	20,127	20,758.0	21,174.7	20,672.7	20,735.2	**20,127.0**	**20,127.0**	20,683.8	20,776.2
d1655	62,128	67,459	71,168.1	63,712.7	64,205.9	63,657.3	64,174.1	**63,615.6**	64,133.7

Better values found using the proposed algorithm are in bold

and 7910. For TSP instances berlin52, eil76, eil101, lin105 and kroA200, pr264, the proposed MGACACO algorithm can find best solutions 7544.37, 539.153, 630.01, 14,381.8, 29,380.2 and 49,138.0, which are close to the best known solutions 7,542,538, 629, 14,379, 29,368 and 49,135. It can, also, be seen from Table 2 that the results of the ACO algorithm are better than the results of the GA, but worse than the results of the PGACS algorithm. That's to say, the PGACS algorithm can obtain better optimization results than GA and ACO algorithm. For larger scale instances, Table 2 shows that the average results of our proposed MGACACO algorithm is better than other three methods. Thus taking into account these results, we could say that the use of the multi-population collaborative strategy and adaptive control parameters is very significant to have improvement in the results. This last issue proves the fact that the use of more than one strategies can significantly improve the results,and the MGACACO algorithm has more exploration abilities.

To further validate the effectiveness of the proposed MGACACO algorithm, the MGACACO algorithm is compared with Pasti's method (Pasti and Castro 2006), Marinakis' method (Marinakisa and Marinaki 2010), and Escario' method (Escario et al. 2015) for 15 TSP benchmark instances from TSPLIB with cities scale from 51 to 1655. The comparison results of the MGACACO algorithm with three other algorithms are shown in Table 3.

As it can be observed from Table 3, the proposed MGACACO algorithm can find best solution for eil51, eil76, rad100, eil101, ch1150, kroB150, kroB200, rat575 and d1655. Escario' method can find best solution for lin105, ch130 kroA200, rat783 and fl140, and Marinakis' method can find best solution for lin318. For TSP instances eil51, rad100 and ch1150, the proposed MGACACO algorithm can find the best known solutions 426 and 7910. For rat783 and fl140, the Escario' method can find the best known solutions 8806 and 20,127. The MGACACO algorithm can find best average solution for eil51, eil76, rad100, eil101, ch1150, kroB150, kroA200, kroB200, rat575 and d1655. Escario' method can find best average solution for lin105, ch130, rat783 and fl140. Marinakis' method can find best average solution for lin318. The MGACACO algorithm can obtain better optimization results than Marinakis' method and Escario' method in most of the given instances by analyzing comparison results.

5.3 The effective analysis of the MGACACO algorithm

According to the results of the simulated experiments from Table 2 and the comparison results of the MGACACO algorithm with Pasti's method, Marinakis' method and Escario' method from Table 3, the proposed MGACACO algorithm can obtain better optimization performance than the GA, ACO, PGACS, Pasti's method, Marinakis' method and Escario' method in solving complex optimization problems in general. Because in the proposed MGACACO algorithm, the GA is used to dynamically adjust the initial pheromone distribution of ACO algorithm to better control the exploitation ability and exploration ability of each subpopulation, and the chaotic optimization method is used to is used to overcome long search time, avoid falling into the local extremum

and improve the search accuracy. The adaptive adjusting pheromone is introduced into each subpopulation to improve the exploration ability and avoid prematurely converging to the local optimal solution. The collaborative strategy is used to dynamically balance the global search ability and local search ability, comprehensively evaluates the state of ACO algorithm and GA, and improves the convergence speed. The simulation results show that the proposed MGACACO algorithm can avoid falling into the local extremum, and takes on better search precision and faster convergence speed.

6 Conclusion

In this paper, a new novel collaborative optimization (MGACACO) algorithm based on the GA, ACO, the chaotic optimization method, multi-population strategy, adaptive control parameters and collaborative strategy was proposed to solve the complex optimization problems. A experimental study has been carried out, using 28 TSP instances and batches of 30 tries, to provide an deep insight in the effect of the proposed MGACACO algorithm. The performance of the proposed MGACACO algorithm is investigated by taking into consideration best and average route length. To show the effectiveness of the MGACACO algorithm, the results of the MGACACO algorithm were compared with the results of the GA, ACO, PGACS, Pasti's method, Marinakis' method and Escario' method. As seen from the experimental results, the proposed MGACACO algorithm gave very good results in most of the given instances and these results are better than the results of the other algorithms in most of the given instances in the comparisons. From the results obtained in this work, it can be concluded that the performance of the proposed MGACACO algorithm is better than or similar to performance of compared methods. In the future, we will focus on the application of the MGACACO algorithm to other stochastic problems, such as Stochastic Vehicle Routing Problem, and in the development of other metaheuristic approaches for solving more and more complex optimization problems.

Acknowledgements The authors would like to thank all the reviewers for their constructive comments. This research was supported by the National Natural Science Foundation of China (U1433124, 51475065), Open Project Program of State Key Laboratory of Mechanical Transmissions (Chongqing University) (SKLMT-KFKT-201416, SKLMT-KFKT-201513), the Natural Science Foundation of Liaoning Province (2015020013), Open Fund of Key Laboratory of Guangxi High Schools for Complex System & Computational Intelligence (15CI06Y), Open Project Program of Guangxi Key laboratory of hybrid computation and IC design analysis (HCIC201507, HCIC201402), Open Project Program of the Traction Power State Key Laboratory of Southwest Jiaotong University (TPL1403), the PAPD fund. The program for the initialization, study, training, and simulation of the proposed algorithm in this article was written with the tool-box of MATLAB 2010b produced by the Math-Works, Inc.

Compliance with ethical standards

Conflict of interest The authors declare that there is no conflict of interest regarding the publication of this manuscript.

References

Akpinar S, Mirac Bayhan G, Baykasoglu A (2013) Hybridizing ant colony optimization via genetic algorithm for mixed-model assembly line balancing problem with sequence dependent setup times between tasks. Appl Soft Comput J 13(1):574–589

Ali RN, Mohammad HF, Niaki STA (2015) A hybrid genetic and imperialist competitive algorithm for green vendor managed inventory of multi-item multi-constraint EOQ model under shortage. Appl Soft Comput J 30(5):353–364

Blum C, Puchinger J, Raidl GR, Roli A (2011) Hybrid metaheuristics in combinatorial optimization: a survey. Appl Soft Comput 11(6):4135–4151

Chen SM, Chien CY (2011) Parallelized genetic ant colony systems for solving the traveling salesman problem. Expert Syst Appl 38(4):3873–3883

Corchado E, Abraham A (2010) Hybrid intelligent algorithms and applications. Inf Sci 180(14):2633–2634

Deng W, Chen R, He B, Liu YQ, Yin LF, Guo JH (2012) A novel two-stage hybrid swarm intelligence optimization algorithm and application. Soft Comput 16(10):1707–1722

Dorigo M, Gambardella LM (1997) Ant colonies for the travelling salesman problem. Biosystems 43(2):73–81

Duarte A, Laguna M, Marti R (2011) Tabu search for the linear ordering problem with cumulative costs. Comput Optim Appl 48(3):697–715

Escario JB, Jimenez JF, Giron-Sierra JM (2015) Ant colony extended: experiments on the travelling salesman problem. Expert Syst Appl 42(1):390–410

Fister I, Perc M, Kamal SM, Fister I (2015) A review of chaos-based firefly algorithms: perspectives and research challenges. Appl Math Comput 252:155–165

Garnier S, Gautrais J, Theraulaz G (2007) The biological principles of swarm intelligence. Swarm Intell 1(1):3–31

Ghanbari A, Kazemi SMR, Mehmanpazir F, Nakhostin MM (2013) A cooperative ant colony optimization-genetic algorithm approach for construction of energy demand forecasting knowledge-based expert systems. Knowl-Based Syst 39(1):194–206

Hamzadayi A, Yildiz G (2013) A simulated annealing algorithm based approach for balancing and sequencing of mixed-model U-lines. Comput Ind Eng 66(4):1070–1084

Holland JH (1977) Adaptation in natural and artificial systems. University of Michigan Press, Ann Arbor

Jing SY (2014) A hybrid genetic algorithm for feature subset selection in rough set theory. Soft Comput 18(7):1373–1382

Jovanovic R, Tuba M (2011) An ant colony optimization algorithm with improved pheromone correction strategy for the minimum weight vertex cover problem. Appl Soft Comput 11(8):5360–5366

Kromer P, Zelinka I, Snasel V (2014) Behaviour of pseudo-random and chaotic sources of stochasticity in nature-inspired optimization methods. Soft Comput 18(4):619–629

Lee ZJ, Su SF, Huang CC, Liu KH (2008) Genetic algorithm with ant colony optimization (GA-ACO) for multiple sequence alignment. Appl Soft Comput J 8(1):55–78

Li WD, Ong SK, Nee AYC (2002) Hybrid genetic algorithm and simulated annealing approach for the optimization of process plans for prismatic parts. Int J Prod Res 40(8):1899–1922

Li BB, Wang L (2007) A hybrid quantum-inspired genetic algorithm for multiobjective flow shop scheduling. IEEE Trans Syst Man Cybern Part B Cybern 37(3):576–591

Mahi M, Baykan ÖK, Kodaz H (2015) A new hybrid method based on particle swarm optimization, ant colony optimization and 3-opt algorithms for travelling salesman problem. Appl Soft Comput J 30(5):484–490

Marinakisa Y, Marinaki M (2010) A hybrid multi-swarm particle swarm optimization algorithm for the probabilistic traveling salesman problem. Comput Oper Res 37(3):432–442

Metaxiotis K, Liagkouras K (2012) Multiobjective evolutionary algorithms for portfolio management: a comprehensive literature review. Expert Syst Appl 39(14):11685–11698

Okamoto T, Hirata H (2013) Global optimization using a multipoint type quasi-chaotic optimization method. Appl Soft Comput 13(2):1247–1264

Otero FEB, Freitas AA, Johnson CG (2013) A new sequential covering strategy for inducing classification rules with ant colony algorithms. IEEE Trans Evol Comput 17(1):64–76

Pan ZQ, Zhang Y, Kwong S (2015) Efficient motion and disparity estimation optimization for low complexity multiview video coding. IEEE Trans Broadcast. doi:10.1109/TBC.2015.2419824

Pasti R, Castro LND (2006) A neuro-immune network for solving the travelling salesman problem. In Proceedings of 2006 international joint conference on neural networks, Vancouver, pp 3760–3766

Rizk-Allah RM, Zaki EM, El-Sawy AA (2013) Hybridizing ant colony optimization with firefly algorithm for unconstrained optimization problems. Appl Math Comput 224:473–483

Robertson BL, Price CJ, Reale M (2013) CARTopt: a random search method for nonsmooth unconstrained optimization. Comput Optim Appl 56(2):291–315

Sheikhan M, Mohammadi N (2012) Neural-based electricity load forecasting using hybrid of GA and ACO for feature selection. Neural Comput Appl 21(8):1961–1970

Sioud A, Gravel M, Gagné C (2012) A hybrid genetic algorithm for the single machine scheduling problem with sequence-dependent setup times. Comput Oper Res 39(10):2415–2424

Tseng LY, Lin YT (2009) A hybrid genetic local search algorithm for the permutation flowshop scheduling problem. Eur J Oper Res 198(1):84–92

Wang XH, Duan HB (2014) A hybrid biogeography-based optimization algorithm for job shop scheduling problem. Comput Ind Eng 73(1):96–114

Wei LY, Zhao M (2005) A niche hybrid genetic algorithm for global optimization of continuous multimodal functions. Appl Math Comput (N Y) 160(3):649–661

Wen Y, Xu H, Yang JD (2011) A heuristic-based hybrid genetic-variable neighborhood search algorithm for task scheduling in heterogeneous multiprocessor system. Inf Sci 181(3):567–581

Wen XZ, Shao L, Xue Y, Fang W (2015) A rapid learning algorithm for vehicle classification. Inf Sci 295(1):395–406

Xing LN, Rohlfshagen P, Chen YW, Yao X (2011) A hybrid ant colony optimization algorithm for the extended capacitated arc routing problem. IEEE Trans Syst Man Cybern Part B Cybern 41(4):1110–1123

Yanass HH (2013) A review of three decades of research on some combinatorial optimization problems. Pesqui Oper 33(1):11–36