View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Old Dominion University

Old Dominion University
ODU Digital Commons

Modeling, Simulation & Visualization Engineering Faculty Publications

Modeling, Simulation & Visualization Engineering

2015

Adaptive Double Chain Quantum Genetic Algorithm for Constrained Optimization Problems

Kong Haipeng

Li Ni

Yuzhong Shen
Old Dominion University, yshen@odu.edu

Follow this and additional works at: https://digitalcommons.odu.edu/msve_fac_pubs

Part of the Aerospace Engineering Commons

Repository Citation

Haipeng, Kong; Ni, Li; and Shen, Yuzhong, "Adaptive Double Chain Quantum Genetic Algorithm for Constrained Optimization Problems" (2015). *Modeling, Simulation & Visualization Engineering Faculty Publications*. 16.
https://digitalcommons.odu.edu/msve_fac_pubs/16

Original Publication Citation

Kong, H., Li, N., & Shen, Y. (2015). Adaptive double chain quantum genetic algorithm for constrained optimization problems. *Chinese Journal of Aeronautics, 28*(1), 214-228. doi:https://doi.org/10.1016/j.cja.2014.12.010

This Article is brought to you for free and open access by the Modeling, Simulation & Visualization Engineering at ODU Digital Commons. It has been accepted for inclusion in Modeling, Simulation & Visualization Engineering Faculty Publications by an authorized administrator of ODU Digital Commons. For more information, please contact digitalcommons@odu.edu.

Chinese Journal of Aeronautics, (2015),28(1): 214–228

Chinese Society of Aeronautics and Astronautics
& Beihang University

Chinese Journal of Aeronautics

cja@buaa.edu.cn
www.sciencedirect.com

Adaptive double chain quantum genetic algorithm for constrained optimization problems

Kong Haipeng [a], **Li Ni** [a,*], **Shen Yuzhong** [b]

[a] *School of Automation Science and Electrical Engineering, Beihang University, Beijing 100191, China*
[b] *Department of Modeling, Simulation and Visualization Engineering, Old Dominion University, Norfolk, VA 23501, USA*

Received 14 January 2014; revised 2 July 2014; accepted 31 October 2014
Available online 26 December 2014

KEYWORDS

Adaptive techniques;
Air combat decision-making;
Constrained optimization;
Multiple target attack;
Quantum genetic algorithm;
Step evolution

Abstract Optimization problems are often highly constrained and evolutionary algorithms (EAs) are effective methods to tackle this kind of problems. To further improve search efficiency and convergence rate of EAs, this paper presents an adaptive double chain quantum genetic algorithm (ADCQGA) for solving constrained optimization problems. ADCQGA makes use of double-individuals to represent solutions that are classified as feasible and infeasible solutions. Fitness (or evaluation) functions are defined for both types of solutions. Based on the fitness function, three types of step evolution (SE) are defined and utilized for judging evolutionary individuals. An adaptive rotation is proposed and used to facilitate updating individuals in different solutions. To further improve the search capability and convergence rate, ADCQGA utilizes an adaptive evolution process (AEP), adaptive mutation and replacement techniques. ADCQGA was first tested on a widely used benchmark function to illustrate the relationship between initial parameter values and the convergence rate/search capability. Then the proposed ADCQGA is successfully applied to solve other twelve benchmark functions and five well-known constrained engineering design problems. Multi-aircraft cooperative target allocation problem is a typical constrained optimization problem and requires efficient methods to tackle. Finally, ADCQGA is successfully applied to solving the target allocation problem.

© 2015 Production and hosting by Elsevier Ltd. on behalf of CSAA & BUAA.

1. Introduction

Most search and optimization problems involve a number of constraints and the constrained optimization technique is a major research area. Generally, the optimal solutions must satisfy two types of constraints including inequality and equality constraints and both can be linear or nonlinear. In this paper, the constrained optimization problem is represented as follows:

$$\min \quad f(\boldsymbol{x})$$
$$\text{s.t.} \begin{cases} g_j(\boldsymbol{x}) \leqslant 0, \ j = 1, 2, \ \ldots, \ J \\ h_k(\boldsymbol{x}) = 0, \ k = 1, 2, \ \ldots, \ K \\ x_i^{\mathrm{l}} \leqslant x_i \leqslant x_i^{\mathrm{u}}, \ i = 1, 2, \ \ldots, \ n \end{cases} \tag{1}$$

* Corresponding author. Tel.: +86 10 82338484.
E-mail address: lini@buaa.edu.cn (N. Li).

Peer review under responsibility of Editorial Committee of CJA.

ELSEVIER Production and hosting by Elsevier

http://dx.doi.org/10.1016/j.cja.2014.12.010
1000-9361 © 2015 Production and hosting by Elsevier Ltd. on behalf of CSAA & BUAA.

where $\boldsymbol{x}$ is the solution vector with size n, and $f(\boldsymbol{x})$ the objective function. There are J lesser-than-equal-to inequality constraints and K equality constraints: $g_j(\boldsymbol{x})$ and $h_k(\boldsymbol{x})$ are the jth inequality and the kth equality constraints, respectively; x_i^{l} and x_i^{u} are the lower and upper bounds of x_i respectively. Additionally, we use Φ to denote the solution space and the feasible solution space is denoted by Ω .

Many approaches have been proposed to address complex constrained optimization problems. Michalewicz and Schoenauer,[1] Eiben,[2] Coello Coello[3] and Salcedo-Sanz[4] discussed various constraint handling methods utilized in evolutionary algorithms (EAs) and categorized the majority of them into five types—penalty functions, repair algorithms, special representations and operators, separate objective and constraints, as well as hybrid methods.

The most common method utilized by EAs is penalty functions, originally proposed by Courant.[5] Its key idea is to incorporate a penalized term into the objective function so that a constrained optimization problem can be transformed into an unconstrained one. Deb[6] pointed that an improper penalty value may cause the algorithm to converge to an infeasible region or some local optimal solutions. Selecting a suitable penalty factor is a difficult issue. Several adaptive penalty techniques have been proposed, such as Coello Coello,[7] Nanakorn and Meesomklin,[8] Barbosa and Lemonge,[9] Farmani and Wright,[10] as well as Wu and Lin.[11] Even though these adaptive penalty methods perform well for some problems, they have additional coefficients that need careful tuning. More recently, an effective co-evolutionary differential evolution for constrained optimization was proposed by Huang et al.[12] Puzzi and Carpinteri[13] presented a double-multiplicative penalty strategy for constrained optimization without the need for penalty factor tuning. However, it is computationally complex and expensive to compute the penalty value.

Coello Coello and Mezura Montes[14] stated that repairing an infeasible individual is to make the infeasible individual feasible. To date, repaired methods have also been widely used to handle constraints. Orvosh and Davis[15,16] reported a so-called 5% rule repair method. In Ref.[15], original individuals are replaced by their ''repaired'' counterparts. Chootinan and Chen[17] combined a gradient-based repair method with genetic algorithms. Salcedo-Sanz[4] reported several popular repair techniques such as crossover operators in permutation encoding and algorithms for fixing the number of 1 s in binary encoded genetic algorithms. In some cases, when an infeasible can be repaired easily, this method is useful. However, in some practical problems, repair operators may introduce a strong bias in the search.[18]

Special representations and operators is another method to tackle constraint problems. The main idea is to preserve the feasible solutions using some special representation schemes. In this method, special operators need to be used due to the change of representation. Bean[19] introduced ''random keys encoding''. Kowalczyk[20] presented the use of constraint consistency. Takahama and Sakai[21] combined constrained method with a differential evolution. Based on the feasible solution number, an adaptive constraint handling technique was put forward by Wang et al.[22] by applying different constraint handling mechanisms to the solutions. Wang and Li[23] utilized the level comparison approach to convert the constrained optimization problem into an unconstrained one. Then the differential evolution algorithm was used to find the optimal solution. Since these methods depend on special design of representation schemes or operators, prior knowledge of the problem is required. However, for some problems, it is difficult or impossible to acquire the prior knowledge.

Dealing with the objective and constraints separately is another type of constraints handling method. A co-evolutionary model was introduced by Paredis,[24] where populations were divided into two groups. These two groups of population evolve at the same time. Powell and Skolnick[25] incorporated a heuristic rule for processing infeasible solutions. In recent years, multi-objective optimization techniques were also utilized. This technique redefined the single-objective optimization problem as a multi-objective one. Venter and Haftka[26] proposed a specialized bi-objective particle swarm optimization algorithm to solve the constrained, single objective optimization problem which is transformed into an unconstrained, bi-objective one.

Combining EAs with other numerical optimization techniques such as Lagrangian multipliers, fuzzy logic or simulated annealing to tackle constraint problems is the main idea of hybrid methods. Adeli and Cheng[27] proposed the augmented Lagrangian genetic algorithm for structural optimization. Fung et al.[28] reported the extension of hybrid genetic algorithm for nonlinear programming problems. Chen and Shahandashti[29] introduced the hybrid of genetic algorithm and simulated annealing for multiple project scheduling with multiple resource constraints. Zhao et al.[30] proposed an effective hybrid genetic algorithm with flexible allowance technique. To make these algorithms work properly, several parameters must be tuned.

Although many effective methods including the ones discussed above have been proposed for various constrained optimization problems, the limitations can be concluded as follows: (A) for penalty functions, it is difficult to tune the penalty factors; (B) the purpose of repair method is to ensure the feasibility of solutions, but it may cause a strong bias toward the repaired local solution space; (C) for some problems, it is difficult if it is impossible to acquire prior knowledge to design the special representations or operators; (D) separation of constraints and objectives is a novel method and its effectiveness needs to be further demonstrated; (E) for hybrid methods, several additional parameters also need to be tuned properly and extra computation is usually needed due to the combination with other optimization techniques.

Quantum computing is a new inter-discipline that combines information science with quantum mechanics. Since the first quantum algorithm[31] was proposed for factoring large prime numbers and another quantum algorithm[32] for searching random databases, quantum algorithms have attracted wide attention. The quantum genetic algorithm (QGA) is a probability optimization algorithm based on quantum computing.[33] Quantum bits (Q-bits) replace general genes, and each individual is constructed with a string of Q-bits. There is no crossover operator in QGA. The quantum gate (Q-gate) is used to update individuals. The search capability of QGA is very efficient as well.[34,35] QGA has rapidly become an international research focus[36–42] because of its unique computing performance. Specifically, Li and Li[43] described the double probability amplitude as the optimal solution chains and the double chain quantum genetic algorithm (DCQGA) was proposed for dealing with continuous optimization problems.

The capability of DCQGA for solving continuous optimization problems has been demonstrated by Li and Li.[43,44] And it was proposed for dealing with continuous optimization problems. Constrained optimization problems are much different from general continuous optimization problems. Constrained optimization problems have two types of constraints, including inequality and equality constraints. And the constraints could be linear or nonlinear. However, DCQGA is not suitable for tackling this type of problem. It lacks constraints handling techniques to obtain optimal solutions. That is why no investigation of constrained optimization problems using DCQGA has been reported. And this is also the motivation of this paper. In this paper, we not only combine DCQGA with our constraints handling technique to tackle constrained optimization problems, but also use special improved techniques to obtain better performance. We named our new algorithm ADCQGA. In ADCQGA, solutions are divided into feasible and infeasible solutions. Step evolution (SE) is proposed to judge evolutionary individuals, based on which an adaptive evolution process (AEP) is devised to improve the search efficiency. Furthermore, in order to improve the search capability and convergence rate of DCQGA, adaptive mutation and replacement technique are developed and utilized.

To assess the effectiveness and efficiency of the ADCQGA, thirteen benchmark functions and five well-known engineering design problems are solved by ADCQGA and its performance is compared against some typical algorithms. ADCQGA is further applied in a typical constrained optimization practice which is Multi-aircraft cooperative target allocation problem. With the increase of detection range of airborne sensor equipment, beyond visual range air combat plays an important role in modern air combat. Multi-aircraft cooperative target allocation, which aims at allocating targets more reasonably under different battlefield situation, is a primary problem in air combat decision-making process. Many scholars have made research into the model of this problem and lots of methods have been proposed.[45–49] Based on the robustness, distributed computation of genetic algorithm and ant colony algorithm, these two algorithms were integrated in Ref.[48] to solve the target assignment problem. Guo et al.[50] utilized crossing and mutating techniques to update particles, and an improved particle swarm algorithm was proposed for air combat decision-making. Niu LW et al. introduced a maximum team performance optimization method to allocate targets.[51] Based on greedy algorithm and ant colony algorithm, Zhang BC et al. put forward an improved algorithm to obtain desired target assignment result.[52] Considering the attack superiority effect on target hit probability, Chen and Wei[53] established a maximum damage model and achieved the target allocation using a particle swarm optimization algorithm. In Ref.[47] cultural algorithm and genetic algorithm were integrated, and the cultural-genetic algorithm was proposed to solve the target allocation problem. In contrast with the aforementioned methods, specific heuristic knowledge was utilized to improve the search capability of optimization algorithms in Refs.[45,49]; meanwhile a heuristic adaptive genetic algorithm (HAGA) and a heuristic ant colony algorithm (HACA) were proposed respectively to solve the target allocation problem. Using heuristic knowledge made HAGA and HACA perform better at the convergence rate than other algorithms mentioned in this literature. However, with the increase of targets and friendly fighters, the searching efficiency of HAGA and HACA would reduce because of the obtaining of heuristic knowledge. For ADCQGA, it is not required to obtain heuristic knowledge, and special techniques can improve its searching capability as well. Therefore, here we apply ADCQGA to tackle the constrained multi-aircraft cooperative target allocation problem and simulation results demonstrate the effectiveness of the proposed algorithm.

The remainder of this paper is organized as follows. Section 2 briefly introduces DCQGA. Section 3 presents the ADCQGA with improved techniques and parameter settings are discussed in Section 4. In Section 5, the proposed algorithm is applied to solving five well-known constrained engineering design problems. Then in Section 6, the proposed algorithm is applied to solving a constrained multi-aircraft cooperative target allocation problem. Finally, conclusions are drawn in Section 7.

2. Double chain quantum genetic algorithm

In quantum computation, the basic information unit is a Q-bit. Dasgupta et al.[54] proposed that the state of a Q-bit can be $|0\rangle, |1\rangle$ or an arbitrary superposition between $|0\rangle$ and $|1\rangle$. It is described as

$$|\varphi\rangle = \alpha|0\rangle + \beta|1\rangle \tag{2}$$

where α and β are state probability amplitudes. $|\alpha|^2$ is the probability that a Q-bit is observed as $|0\rangle$, and similarly $|\beta|^2$ is the probability that a Q-bit is observed as $|1\rangle$. Additionally, they have to satisfy the normalization constraint:[54]

$$|\alpha|^2 + |\beta|^2 = 1. \tag{3}$$

As described in Eqs. (2) and (3), a Q-bit can be represented by $[\alpha, \beta]^{\mathrm{T}}$,[43] where each Q-bit is regarded as a pair of coordinates.

The DCQGA encodes individuals using the probability amplitudes.[43] Taking Eq. (3) into account, the encoding scheme is described as

$$\boldsymbol{P}_t = \begin{bmatrix} \cos r_{t1} & \cos r_{t2} & \cdots & \cos r_{ti} & \cdots & \cos r_{tn} \\ \sin r_{t1} & \sin r_{t2} & \cdots & \sin r_{ti} & \cdots & \sin r_{tn} \end{bmatrix} \tag{4}$$

where $t = 1, 2, \ldots, m$; $i = 1, 2, \ldots, n$; m is the population size, and n the size of Q-bits (that is the number of unknowns), $\boldsymbol{P}_t$ the tth double-individual, and r_{ti} the ith quantum angle of $\boldsymbol{P}_t$, with $r_{ti} = 2\pi\mathrm{rand}(\cdot), \mathrm{rand} \in [0, 1]$. Every double-individual is composed of two parallel gene chains which are sine solution and cosine solution.[43]

$$\boldsymbol{P}_t = \begin{bmatrix} \boldsymbol{p}_{tc} \\ \boldsymbol{p}_{ts} \end{bmatrix} \tag{5}$$

where

$$\begin{cases} \boldsymbol{p}_{tc} = [\cos(r_{t1}) | \cos(r_{t2}) | \cdots | \cos(r_{tn})] \\ \boldsymbol{p}_{ts} = [\sin(r_{t1}) | \sin(r_{t2}) | \cdots | \sin(r_{tn})] \end{cases} \tag{6}$$

where $\boldsymbol{p}_{tc}$ and $\boldsymbol{p}_{ts}$ are the sine solution and cosine solution of the tth double-individual, respectively. Thus, the two chains update simultaneously with the quantum angle.

Fig. 1 illustrates the flowchart of DCQGA. It can be seen that there are three main steps (solution space mapping, Q-

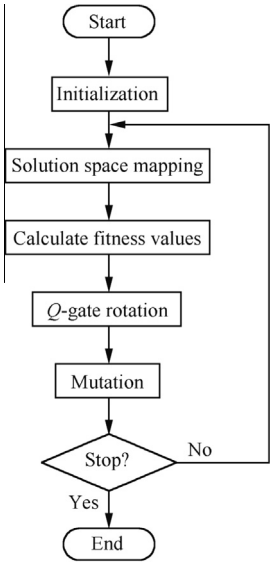


Fig. 1 Flowchart of DCQGA.

gate rotation and mutation) in DCQGA to obtain the optimal solution.

(1) Solution space mapping. Because all gene values are trigonometric function values in DCQGA and all unknowns of optimization problems described in Eq. (1) are constrained between lower and upper bounds, linear transformation is used to transform trigonometric function values into the solution space. The linear transformation formula is as follows:[43]

$$\begin{cases} x_{ic} = x_i^{\mathrm{l}} + \dfrac{\cos r_i + 1}{2}(x_i^{\mathrm{u}} - x_i^{\mathrm{l}}) \\ x_{is} = x_i^{\mathrm{l}} + \dfrac{\sin r_i + 1}{2}(x_i^{\mathrm{u}} - x_i^{\mathrm{l}}) \end{cases} \tag{7}$$

Because a double-individual has two genes, x_{ic} and x_{is} are used to represent the cosine and sine values (multiplied by the magnitude) of the ith unknown. r_i is the ith quantum angle of an double-individual and the boundary of trigonometric function is $[-1,1]$.

(2) Q-gate rotation. The DCQGA updates the probability amplitudes by rotating the quantum angles with a Q-gate. Denote the rotation angle as θ and the Q-gate[43] is described as

$$\mathrm{Q-gate} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \tag{8}$$

The rotation process is[43]

$$\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} \cos r_i \\ \sin r_i \end{bmatrix} = \begin{bmatrix} \cos(r_i + \theta) \\ \sin(r_i + \theta) \end{bmatrix} \tag{9}$$

(3) Q-gate mutation. The individuals mutate Q-bits using a quantum NOT-gate. NOT-gate[43] is a special Q-gate with a rotation angle of $\pi/2$. Setting θ to $\pi/2$ in Eq. (8), the probability amplitudes of the mutation Q-bits will be swapped.

3. Adaptive double chain quantum genetic algorithm

3.1. Evaluation functions

Both feasible and infeasible solutions have their own evaluation functions. The fitness function for the feasible solution is the objective function. For the infeasible solution, it is meaningless to calculate its objective function value due to its constraint violation. However, the constraint violation is still a "good or bad" reflection of infeasible solutions. At least, we can make use of the constraint violation to guide these infeasible solutions to the feasible region. Wang and Yin[55] calculated the sum of violated constraints as follows:

$$G(\boldsymbol{x}) = \sum_{j=1}^{J} \max\{0, g_j(\boldsymbol{x})\} + \sum_{k=1}^{K} |h_k(\boldsymbol{x})| \tag{10}$$

Thus, the fitness of feasible or infeasible solutions can be determined as follows:

$$\text{fitness} = \begin{cases} f(\boldsymbol{x}), & \boldsymbol{x} \in \Omega \\ G(\boldsymbol{x}), & \boldsymbol{x} \notin \Omega \end{cases} \tag{11}$$

3.2. Step evolution

As its name suggests, an evolutionary algorithm is essentially a gradual evolution process. Considering the constraint problem, we define three types of step evolutions.

(1) G-SE: for an individual of generation v that is an infeasible solution with fitness $G_v(\boldsymbol{x})$, if the individual after updating is still an infeasible solution but with better fitness $G'_v(x)$ (i.e., $G'_v(\boldsymbol{x}) < G_v(\boldsymbol{x})$), we call the updating process G-SE.
(2) GF-SE: for an individual of generation v that is an infeasible solution, if the individual after updating becomes a feasible solution, we call the updating process GF-SE.
(3) F-SE: for an individual of generation v that is a feasible solution with fitness $f_v(\boldsymbol{x})$, if the individual after updating is also a feasible solution but with better fitness $f'_v(\boldsymbol{x})$ (i.e., $f'_v(\boldsymbol{x}) < f_v(\boldsymbol{x})$), we call the updating process F-SE.

Note that this paper only considers positive evolutions (G-SE, GF-SE, F-SE) that produce better solutions in terms of fitness values. This paper will develop the rotation angle formula and adjustable mutation formula to converge the individuals to the optimal value.

3.3. Rotation angle

Similar to DCQGA,[44] ADCQGA updates the probability amplitudes by rotating the quantum angles with Q-gates. The rotation angle is computed as follows:[44]

$$\begin{aligned} &\theta_{ti} = -\mathrm{sgn}A\theta_0 \exp\left(-\frac{|\nabla f(x_{ti})| - \nabla f_{i\min}}{\nabla f_{i\max} - \nabla f_{i\min}}\right) \\ &\text{s.t.} \quad A = \begin{vmatrix} \alpha_{bi} & \alpha_{ti} \\ \beta_{bi} & \beta_{ti} \end{vmatrix} \end{aligned} \tag{12}$$

where

$$\begin{cases} \nabla f(x_{ti}) = \dfrac{\partial f(\boldsymbol{x}_t^{\mathrm{b}})}{\partial x_{ti}} \\ \nabla f_{i\max} = \max\left\{ \left|\dfrac{\partial f(\boldsymbol{x}_1^{\mathrm{b}})}{\partial x_{1i}}\right| , \left|\dfrac{\partial f(\boldsymbol{x}_2^{\mathrm{b}})}{\partial x_{2i}}\right|, \cdots , \left|\dfrac{\partial f(\boldsymbol{x}_m^{\mathrm{b}})}{\partial x_{mi}}\right| \right\} \\ \nabla f_{i\min} = \min\left\{ \left|\dfrac{\partial f(\boldsymbol{x}_1^{\mathrm{b}})}{\partial x_{1i}}\right| , \left|\dfrac{\partial f(\boldsymbol{x}_2^{\mathrm{b}})}{\partial x_{2i}}\right|, \cdots , \left|\dfrac{\partial f(\boldsymbol{x}_m^{\mathrm{b}})}{\partial x_{mi}}\right| \right\} \end{cases} \tag{13}$$

where $t = 1, 2, \ldots, m$; $i = 1, 2, \ldots, n$; t represents the number of double-individuals; θ_{ti} denotes the rotation angle of the ith Q-bit, θ_0 the initial rotation angle; $[\alpha_{ti}, \beta_{ti}]^{\mathrm{T}}$ and $[\alpha_{bi}, \beta_{bi}]^{\mathrm{T}}$ are the probability amplitudes of Q-bit i of double-individual t and the current global best double-individual respectively; sgn$A$ is the sign of determinant A. Additionally, x_{ti} denotes the ith gene of the better gene chain. The better gene chain of double-individual t, denoted by $\boldsymbol{x}_t^b$, means the better one (with smaller fitness) between cosine solution and sine solution. $\nabla f(x_{ti})$ is the partial derivative of fitness function $f(\boldsymbol{x})$ with respect to x_{ti}. $\nabla f_{i\max}$ and $\nabla f_{i\min}$ are the maximum and minimum partial derivatives of fitness function $f(\boldsymbol{x})$ among all double-individuals of current generation with respect to x_{ti}. For discrete optimization problems, because of the inexistence of derivatives, first order differences are used to replace the partial derivatives. $\nabla f(x_{ti})$, $\nabla f_{i\max}$ and $\nabla f_{i\min}$ are computed as follows:[44]

$$\begin{cases} \nabla f(x_{ti}) = f(x_{ti}^{\mathrm{p}}) - f(x_{ti}^{\mathrm{c}}) \\ \nabla f_{i\max} = \max\{|f(x_{1i}^{\mathrm{p}}) - f(x_{1i}^{\mathrm{c}})|, \cdots, |f(x_{mi}^{\mathrm{p}}) - f(x_{mi}^{\mathrm{c}})|\} \\ \nabla f_{i\min} = \min\{|f(x_{1i}^{\mathrm{p}}) - f(x_{1i}^{\mathrm{c}})|, \cdots, |f(x_{mi}^{\mathrm{p}}) - f(x_{mi}^{\mathrm{c}})|\} \end{cases} \tag{14}$$

where $t = 1, 2, \ldots, m; i = 1, 2, \ldots, n$; x_{ti}^{p} and x_{ti}^{c} denote the ith parent and child gene of the better gene chain of double-individual t.

For constrained optimization problems, the angle rotation defined in Eq. (12) cannot be directly applied because of the diversity of fitness functions. In this case, we compute it as follows:

$$\theta_{ti} = \begin{cases} -\mathrm{sgn}A\theta_0 \exp(-\dfrac{|\nabla f(x_{ti})| - \nabla f_{i\min}}{\nabla f_{i\max} - \nabla f_{i\min}}), & \boldsymbol{x}_t^{\mathrm{b}} \in \Omega \\ -\mathrm{sgn}A\theta_0 \exp(-\dfrac{|\nabla G(x_{ti})| - \nabla G_{i\min}}{\nabla G_{i\max} - \nabla G_{i\min}}), & \boldsymbol{x}_t^{\mathrm{b}} \notin \Omega \end{cases} \tag{15}$$

If the better gene chain of double-individual t is a feasible solution, the rotation angle is computed in the same way as DCQGA. Otherwise, the constraint violation function replaces the objective function in computing the rotation angle. It should be noted that the definitions of maximum and minimum partial derivatives of fitness are different in terms of the fitness functions. ∇f_imax and ∇f_imin are the maximum and minimum partial derivatives of fitness function $f(\boldsymbol{x})$ with respect to x_{ti} among all feasible double-individuals of current generation. Similarly, ∇G_imax and ∇G_imin are the maximum and minimum partial derivatives of constraint violation function $G(\boldsymbol{x})$ with respect to x_{ti} among all the infeasible double-individuals.

3.4. Adaptive evolution process

In order to improve the efficiency of the proposed algorithm, we refine the individual updating process. Section 3.2 classified the step evolution into three types: G-SE, GF-SE and F-SE. If the fitness of any individual in the population after updating is better than the previous global optimal solution, the updating process is SE, and these individuals are called step evolution individuals (SEIS).

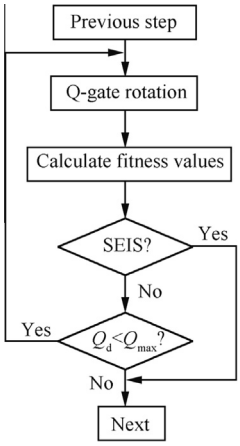


Fig. 2 Adaptive evolution process.

With the definition of SEIS, we propose an adaptive evolution process as shown in Fig. 2. Although Q-gate rotations are used by both DCQGA and ADCQGA to update individuals, the specific updating process in ADCQGA is different. In ADCQGA, the first step is to perform Q-gate rotation. Then fitness values are calculated. Before proceeding to the next step, individuals are evaluated first. If the individuals are SEIS, they exit the AEP. Otherwise, they need to update continually. To prevent an infinite updating loop, AEP utilizes a maximum number of rotations that is allowed for each individual, denoted by $Q_{\max}$, and when $Q_{\max} = 1$, there will be no difference between DCQGA and ADCQGA. In order to make AEP play a role in ADCQGA, we should set $Q_{\max} > 1$. Usually, individuals of one generation undergo Q-gate rotation a few times before they proceed to the next step, and the total number of iterations is called Q-gate rotation degree, denoted by Q_{d}.

In DCQGA, individuals rotate only once in every updating process. For ADCQGA, if individuals are SEIS after one rotation, then the updating process is the same as DCQGA. For some optimization problems, flat solution regions are common. Because of the small rotation angle, DCQGA would need several evolution generations to step through the flat region. ADCQGA is more efficient in handling such cases because ADCQGA will perform Q-gate rotation several times in one generation, resulting in an overall larger rotation angle in one generation and thus fewer evolution generations in the same flat region.

3.5. Mutation

The mutation probability of DCQGA is a constant initialized at the beginning of the whole evolutionary process. It is difficult to select a universal mutation probability applicable to different optimization problems. To address this problem, we propose the following adaptive mutation approach.

The Q-gate rotation degree denotes the number of iterations that the individuals undergo in AEP and it can be used to compute the mutation probability as follows:

$$p_{tm} = \begin{cases} \dfrac{Q_{\mathrm{d}}}{Q_{\max}} p_{m0} \exp\left(-\dfrac{|f_{\max} - f(\boldsymbol{x}_t^{\mathrm{b}})|}{|f_{\max} - f_{\min}|}\right), & \boldsymbol{x}_t^{\mathrm{b}} \in \Omega \\ \dfrac{Q_d}{Q_{\max}} p_{m0} \exp\left(-\dfrac{|G_{\max} - G(\boldsymbol{x}_t^{\mathrm{b}})|}{|G_{\max} - G_{\min}|}\right), & \boldsymbol{x}_t^{\mathrm{b}} \notin \Omega \end{cases} \tag{16}$$

where $t = 1, 2, \cdots, m$; p_{tm} denotes the mutation probability of double-individual t, p_{m0} the initial mutation probability, $\boldsymbol{x}_t^{\mathrm{b}}$ the better gene chain between cosine solution and sine solution of double-individual t; $f(\boldsymbol{x}_t^{\mathrm{b}})$ and $G(\boldsymbol{x}_t^{\mathrm{b}})$ denote the objective function value and constraint violation function value of $\boldsymbol{x}_t^{\mathrm{b}}$, respectively; $f_{\max}$ and $f_{\min}$ are the current global maximum and minimum objective function values, respectively. Similarly $G_{\max}$ and $G_{\min}$ are the current global maximum and minimum constraint violation function values. Each pair of double-individuals in the same generation share the same Q_{d}. From Eq. (16), we can find out that the mutation probability changes in accordance with Q_{d} and fitness values.

From Eq. (16) we can see that the worse the fitness of current individuals, the bigger p_{tm} would be. On the other hand, in a flat region, if $Q_{\mathrm{d}} = Q_{\max}, p_{tm}$ would not be influenced by $Q_{\mathrm{d}}/Q_{\max}$. Otherwise, if $Q_{\mathrm{d}} < Q_{\max}$, it means the individuals of current generation is SEIS. Therefore p_{tm} would be rather smaller because the current individuals are getting better. Moreover, the bigger Q_{d} is, the bigger p_{tm} would be.

The mutation is implemented as follows:

(1) First generate a random number between 0 and 1 for every double-individual.
(2) For each double-individual, if the generated number is smaller than p_{tm}, then go to (3), otherwise exit the mutation process.
(3) Choose a Q-bit randomly from the double-individual and swap the probability amplitudes of this Q-bit.

3.6. Replacement

Many researchers have used repaired methods to improve the convergence rates and solution quality of optimization algorithms as discussed in Section 1. Repaired methods intend to change an infeasible individual to a feasible one. But in order to repair infeasible individuals, special functions or a priori knowledge is needed. Moreover, the repaired solutions may not be better than the current solutions already found. In this paper, we use "replacement methods" to improve our optimization algorithm.

First, if both the individuals of a double-individual are infeasible solutions, we name the double-individual as an infeasible double-individual. For all infeasible double-individuals in a generation, they all have a probability to be replaced by the current global best solution. If the current best global solution is a feasible one, it does not need to repair infeasible solutions through special functions or priori knowledge, because a simple replacement of the infeasible solution with the global best solution repairs it.

Not all infeasible solutions would be replaced and the replacement probability is computed as follows:

$$p_{tr} = p_{r0} \frac{m_G}{m} \exp\left(-\frac{|G_{\max} - G(\boldsymbol{x}_t^{\mathrm{b}})|}{G_{\max} - G_{\min}}\right) \tag{17}$$

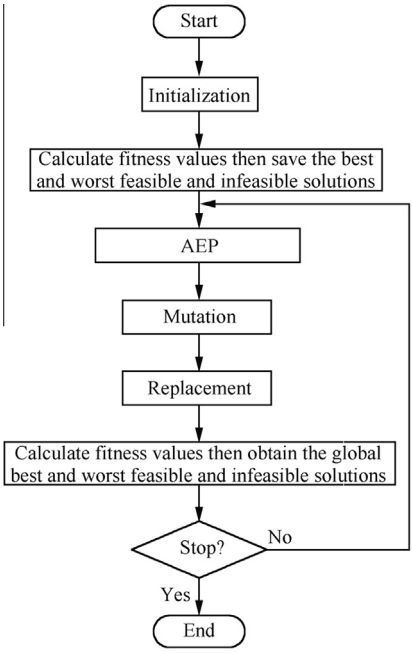


Fig. 3 Flowchart of ADCQGA.

where $t = 1, 2, \cdots, m$; p_{tr} denotes the replacement probability of a double-individual t, p_{r0} the initial replacement probability, m the population size, and m_G the number of infeasible double-individual (if $\boldsymbol{x}_t^b$ is an infeasible solution, then double-individual t is an infeasible double-individual).

From Eq. (17) we can see that p_{tr} is related with $G(\boldsymbol{x}_t^{\mathrm{b}})$ and m_G. The worse constraint violation function value of the double-individual is, the bigger p_{tr} would be. Meanwhile, the more the number of infeasible double-individuals is, the greater p_{tr} would be as well.

The replacement is implemented as follows:

(1) First generate a random number between 0 and 1 for every infeasible double-individual.
(2) For each infeasible double-individual, if the generated number is smaller than its replacement probability, then go to (3), otherwise exit the replacement process.
(3) Replace the infeasible double-individual with the current global best solution.

To summarize ADCQGA, Fig. 3 illustrates the entire flowchart of ADCQGA.

Step 1. Population initialization. Generate m double-individuals to form the initial population according to Eq. (4).
Step 2. Transform all genes from unit space $I^n = [-1, 1]^n$ to the solution space Φ. Then calculate fitness values of all individuals and save the best and worst feasible and infeasible solutions.
Step 3. AEP.
Step 4. Mutation.
Step 5. Replacement.
Step 6. Calculate fitness values again then obtain the global best and worst feasible and infeasible solutions as Step 2.
Step 7. If the best feasible solution satisfies the termination condition, ADCQGA ends. Otherwise, it goes back to Step 3.

4. Setting of experimental parameters

The applicability and effectiveness of most algorithms for constrained optimization problems are demonstrated by benchmark functions.[12,14,22,23,30,42,56] In the same way, to evaluate the performance of the ADCQGA, we use the thirteen benchmark functions described in Ref.[56].

For ADCQGA, the parameters include θ_{ti}, p_{tm} and p_{tr}. These parameters are all adaptively updated in the evolution process, but their initial values need to be set manually, which are $\theta_0, p_{m0}, Q_{\max}$ and p_{r0}. θ_0 is also a parameter of DCQGA that has been discussed in Refs.[43,44]. In this paper, we still set $\theta_0 = 0.001\pi$ as DCQGA did.

g9 is one of the thirteen well-known benchmark functions in Ref.[56]. Additionally, g9 is a typical benchmark function with nonlinear objective function and 4 nonlinear inequalities. In order to analyze the relationship between the other parameters ($Q_{\max}, p_{m0}$ and p_{r0}) and the convergence rate/search capability, first we applied ADCQGA to g9. Then other benchmark functions are solved with the discussed parameter values.

Eq. (18) shows the definition and constraints of g9.

$$f(\boldsymbol{x}) = (x_1 - 10)^2 + 5(x_2 - 12)^2 + x_3^4 + 3(x_4 - 11)^2 + 10x_5^6 + 7x_6^2 + x_7^4 - 4x_6x_7 - 10x_6 - 8x_7$$

s.t.

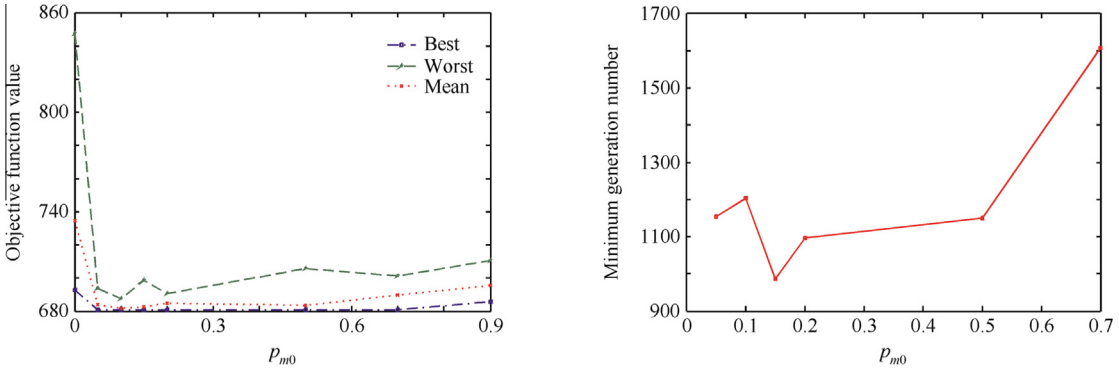
$$\begin{cases} g_1(\boldsymbol{x}) = -127 + 2x_1^2 + 3x_2^4 + x_3 + 4x_4^2 + 5x_5 \leqslant 0 \\ g_2(\boldsymbol{x}) = -282 + 7x_1 + 3x_2 + 10x_3^2 + x_4 - x_5 \leqslant 0 \\ g_3(\boldsymbol{x}) = -196 + 23x_1 + x_2^2 + 6x_6^2 - 8x_7 \leqslant 0 \\ g_4(\boldsymbol{x}) = 4x_1^2 + x_2^2 - 3x_1x_2 + 2x_3^2 + 5x_6 - 11x_7 \leqslant 0 \end{cases} \tag{18}$$

where $-10 \leqslant x_i \leqslant 10 (i = 1, 2, \cdots, 7)$. The optimal solution is $\boldsymbol{x}$ = [2.330499, 1.951372, −0.4775414, 4.365726, −0.624487, 1.038131, 1.594227], where $f(\boldsymbol{x}) = 680.6300573$.

(1) Firstly, the range of p_{m0} is investigated. The population size is 30 and $p_{r0} = 0$. Here, we only investigate the influence of p_{m0} and set $Q_{\max} = 1$ to avoid the influence of AEP. When $p_{m0} = \{0, 0.05, 0.1, 0.15, 0.2, 0.5, 0.7, 0.9\}$, 30 independent runs were performed for each experiment. The optimization results are presented in Fig. 4(a), it shows the mean, best, and worst objective function values of the 30 independent runs for different p_{m0} values. It can be seen when $0.05 \leqslant p_{m0} \leqslant 0.7$, the optimal solution can be obtained, and among all these successful experiments, the minimum generation numbers that have successfully obtained the optimum solution are shown in Fig. 4(b), showing that when $p_{m0} = 0.15$, the algorithm can find the optimal solution less than 1000 generations, which is the smallest among all the experiments.

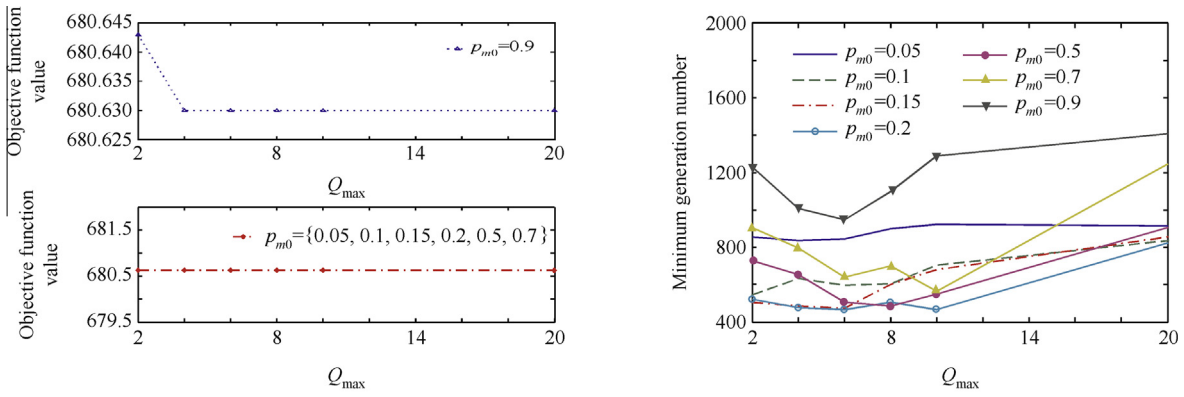
(2) Secondly, the range of $Q_{\max}$ is investigated. The population size is also 30 and the replacement probability is 0. Because of the relationship between p_{m0} and $Q_{\max}$ shown in Eq. (16), different experiments are tested respectively for $p_{m0} = \{0.05, 0.1, 0.15, 0.2, 0.5, 0.7, 0.9\}$. For $Q_{\max} = \{2, 4, 6, 8, 10, 20\}$, 30 independent runs were performed for each experiment as well, and the best optimization results under every p_{m0} are shown in Fig. 5(a), we can see that all the experiments can obtain the optimal solution when $Q_{\max} = 2$ except when $p_{m0} = 0.9$. But for $Q_{\max} > 2$, ADCQGA can find the optimal solution even when $p_{m0} = 0.9$. It means AEP improves the search capability of ADCQGA with a wide range of mutation probability. Furthermore, the minimum generation numbers that have successfully obtained the optimum solution are also shown in Fig. 5(b), it can be seen that the minimum generation numbers are reduced because of the use of AEP. In particular, when $0.05 < p_{m0} < 0.9$ and $4 \leqslant Q_{\max} \leqslant 6$, AEP can accelerate the optimization process better than other conditions. The results shown in Figs. 4 and 5 demonstrate that ADCQGA with parameters $0.1 < p_{m0} < 0.7$ and $4 \leqslant Q_{\max} \leqslant 6$ produces optimal results. For the remaining tests, we set $p_{m0} = 0.2, Q_{\max} = 5$.

(3) Finally, the replacement probability p_{r0} is investigated. The population size is also 30. For $p_{r0} = \{0.05, 0.1, 0.15, 0.2, 0.5, 0.7, 0.9\}$, 30 independent runs are performed for each experiment, and the optimization results are shown in Fig. 6(a), it can be seen that the optimal solution can be found when $p_{r0} = \{0.05, 0.1, 0.15, 0.2, 0.5, 0.7, 0.9\}$. Furthermore, the mean function value and the worst function value are all better than the results without using p_{r0} shown in Fig. 4(a). The minimum generation numbers that have successfully obtained the optimum solution are shown in Fig. 6(b). It can been seen that when $0.1 < p_{r0} < 0.7$, the replacement technique can accelerate the optimization process and the minimum generation numbers are all less than 400, especially when $p_{r0} = 0.2$.



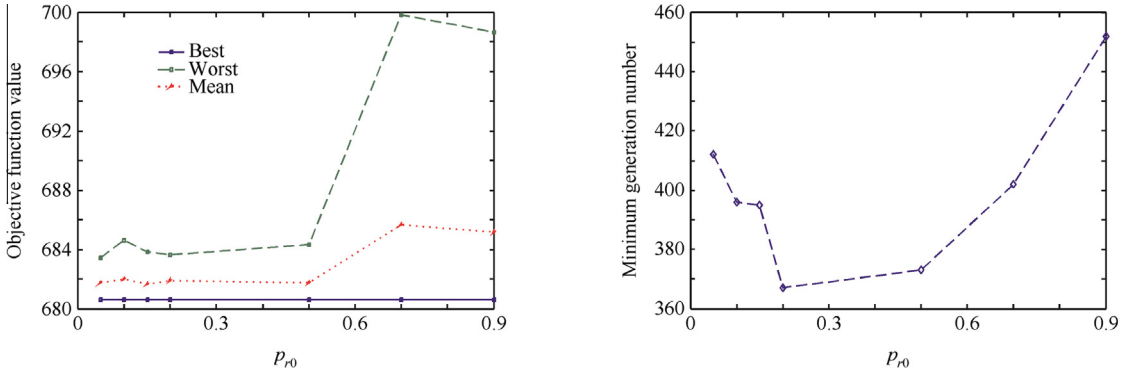
(a) Optimization results (b) Minimum generation numbers

Fig. 4 Optimization results of the ADCQGA under different p_{m0}.



(a) Optimization results (b) Minimum generation numbers

Fig. 5 Objective function values and minimum generation numbers under different $Q_{\max}$ and p_{m0}.



(a) Optimization results (b) Minimum generation numbers

Fig. 6 Optimization results of the ADCQGA under different p_{r0}.

Based on all the experiments discussed above, we can see that when $0.1 < p_{m0} < 0.7, 4 \leqslant Q_{\max} \leqslant 6$ and $0.1 < p_{r0} < 0.7$, both the search capability and convergence rate of ADCQGA are good. Because the mutation probability and the replacement probability are adaptively updated in the evolution process, so the good performance of ADCQGA can be obtained with a wide range of initial parameter values.

Then based on the results discussed above, we set $\theta_0 = 0.001\pi, p_{m0} = 0.2, p_{r0} = 0.2, Q_{\max} = 5$, with 50 independent runs; we apply ADCQGA to solve other 12 benchmark functions in Ref.[56]. To further verify the performance of ADCQGA, comparisons are carried out with four typical algorithms from the literatures, including ISR,[57] SMES,[58] HEAA[22] and GAFAT.[30] Table 1 shows the best values obtained by these comparative algorithms. From Table 1 we can find out that for g1, g3, g4, g6, g7, g8, g9 g11 and g12, ADCQGA performs better or as well as other algorithms. For the other 4 functions, the best results obtained by ADCQGA are a little

Table 1 Comparisons of these comparative algorithms.

Benchmark function	ADCQGA	ISR	SMES	HEAA	GAFAT
g1	−15.000000	−15.000000	−15.000000	−15.000000	−15.000000
g2	0.8035200	0.8036190	0.8036010	0.8035820	0.8031730
g3	1.0000000	1.0000000	1.0000000	1.0000000	1.0000000
g4	−30665.539	−30665.539	−30665.539	−30665.539	−30665.539
g5	5126.4982	5126.4970	5126.5990	5126.4980	5126.4980
g6	−6961.8140	−6961.8140	−6961.8140	−6961.8140	−6961.8140
g7	24.306000	24.306000	24.327000	24.306000	24.306000
g8	0.0958250	0.0958250	0.0958250	0.0958250	0.0958250
g9	680.63000	680.63000	680.63000	680.63000	680.63000
g10	7049.3400	7049.2480	7051.9030	7049.2480	7049.2480
g11	0.7500000	0.7500000	0.7500000	0.7500000	0.7500000
g12	1.0000000	1.0000000	1.0000000	1.0000000	1.0000000
g13	0.0539793	0.0539420	0.0539860	0.0539498	0.0539498

worse than the results obtained by others. In conclusion, ADCQGA is a competitive algorithm for constrained optimization problems.

5. Experiments on constrained engineering design problems

To assess the effectiveness and efficiency of the ADCQGA, five well-known engineering design problems were solved by ADCQGA and its performance was compared against that of other algorithms, including DCQGA,[44] genetic algorithm (GA)[59] and others. Table 2 illustrates the parameter values of these algorithms.

5.1. Case 1: spring design

The spring design problem is to solve for the minimum weight of a spring subject to constraints on minimum deflection, surge frequency, shear stress, and limits on outside diameter. This problem involves three continuous variables and four nonlinear inequality constraints. It has been studied by Coello Coello and Becerra,[60] Huang et al.,[12] Zhang et al.[42] and Wang and Li.[23] The best results of different algorithms are shown in Table 3 from which it can be seen that ADCQGA achieved the best objective function value. In particular, the best function value obtained by DCQGA is 0.012740377, worse than that of ADCQGA.

To compare GA, DCQGA and ADCQGA in detail, Fig. 7 shows the best convergence processes of these three algorithms in 50 independent runs. It takes about 270 generations for ADCQGA to converge to the minimum fitness function value of 0.012665233, compared to 900 generations needed by DCQGA to obtain a fitness value of 0.012740377. GA had to undergo more generations for a larger fitness value.

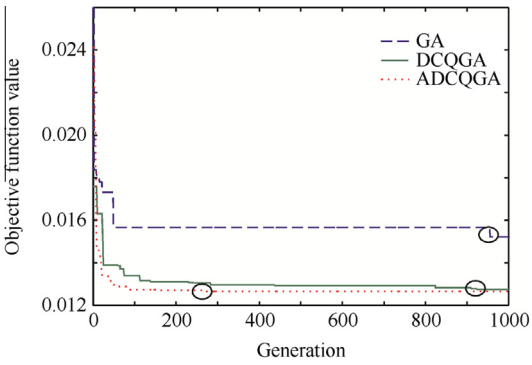


Fig. 7 Best convergence processes for Case 1.

5.2. Case 2: welded beam design 1

Welded beam design is a well-known optimization problem. The goal is to minimize the cost of the beam subject to constraints on shear stress, bending stress, buckling load and the end deflection. Some algorithms have been applied to this problem such as Ragsdell and Phillips,[61] Deb,[62] Ray and Liew,[63] Wang and Yin,[55] and Wang and Li.[23] The best results are shown in Table 4 from which it can be seen that the best fitness function value was obtained by ADCQGA and Wang and Li's method.

Furthermore, Fig. 8 illustrates the best convergence processes of GA, DCQGA and ADCQGA in 50 independent runs. The value 2.3809565803 was obtained by ADCQGA in less than 450 generations, compared to 900 generations needed by DCQGA for a fitness value of 2.5610110253. A fitness value of 3.0964676765 was obtained by GA in about 580 generations.

5.3. Case 3: welded beam design 2

This kind of welded beam design is another well-studied optimization problem. To satisfy the constraints of shear stress, bending stress, buckling load, end deflection, and side constraint, the design is to find the minimum fabricating cost of the welded beam. Some algorithms have been proposed to solve this problem such as Coello Coello and Mezura Montes,[14] Coello Coello and Becerra,[60] Huang et al.,[12] Mezura-Montes and Coello,[64] and Wang and Li.[23] Table 5 lists the best results.

From Table 5 it can be seen that the best solution obtained by ADCQGA is much better than other algorithms. The best

Table 2 Setting of parameters.

Parameter	GA	DCQGA	ADCQGA
Initial rotation angle (°)		0.001π	0.001π
Initial mutation probability			0.2
Initial replacement probability			0.2
$Q_{\max}$			5
Mutation probability	0.15	0.05	
Crossover probability	0.8		
Population size	1000	500	100
Maximum generation number	1000	1000	1000

Table 3 Comparison of the best results of Case 1.

Algorithm	x_1	x_2	x_3	$f(\boldsymbol{x})$
Ref.[60]	0.3173950000	0.0500000000	14.0317950000	0.012721000
Ref.[12]	0.3547140000	0.0516090000	11.4108310000	0.012670200
Ref.[42]	0.3567177469	0.0516890614	11.2889653382	0.012665233
Ref.[23]	0.3567177413	0.0516890611	11.2889656626	0.012665233
GA	0.5758980167	0.0595948170	5.4421330798	0.015221589
DCQGA	0.3998746612	0.0534364839	9.1579045190	0.012740377
ADCQGA	0.3567177390	0.0516890609	11.2889658089	0.012665233

Table 4 Comparison of the best results of Case 2.

Algorithm	x_1	x_2	x_3	x_4	$f(x)$
Ref.[61]	0.2455000000	6.1960000000	8.2730000000	0.2455000000	2.3859370000
Ref.[62]	0.2489000000	6.1730000000	8.1789000000	0.2533000000	2.4331160000
Ref.[63]	0.2444382760	6.2379672340	8.2885761430	0.2445661820	2.3854347000
Ref.[55]	0.2443689800	6.2175197100	8.2914714000	0.2443689800	2.3809565817
Ref.[23]	0.2443689758	6.2175197152	8.2914813905	0.2443689758	2.3809565800
GA	0.2603451923	9.3147686010	8.0151245351	0.2668424048	3.0964676765
DCQGA	0.1968240405	8.0464042896	8.5912098139	0.2432600836	2.5610110253
ADCQGA	0.2443689758	6.2175197152	8.2914813905	0.2443689758	2.3809565803

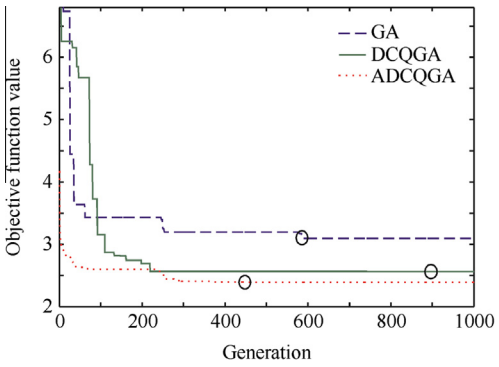


Fig. 8 Best convergence processes for Case 2.

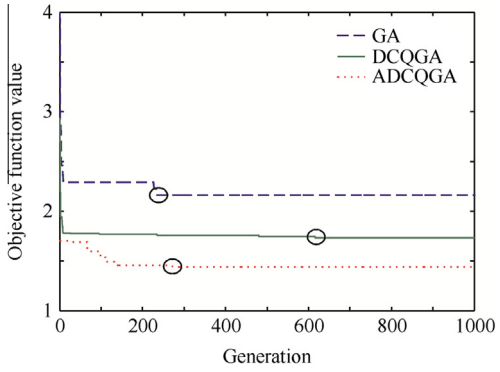


Fig. 9 Best convergence processes for Case 3.

objective function value obtained by DCQGA is 1.730715, which is also better than that of Huang's and Monte's. Fig. 9 shows the best convergence processes of GA, DCQGA and ADCQGA in 50 independent runs. The ADCQGA achieved the best solution in the less than 300 generations, about half of the 600 generations needed by DCQGA for an objective function value of 1.730715. Even though GA converged in less than 300 generations, its objective function value (2.161399) is the largest among all the algorithms shown in Table 5.

5.4. Case 4: speed reducer design

To satisfy the constraints of bending stress of gear teeth, surface stress, transverse deflections of a shaft and stress in the shaft, the weight of a speed reducer needs to be minimized. Some algorithms have been proposed to solve this problem (Ray and Liew,[63] Mezura-Montes et al.,[65] Zhang et al.,[42] Wang and Li[23]).

Table 6 lists the best results of these algorithms. Fig. 10 illustrates the best convergence process of GA, DCQGA and ADCQGA in 50 independent runs. It can be seen that the best objective function value obtained by ADCQGA in 420 generations is 2992.184537, which is better than others, while for DCQGA, for about 200 generations the value 3003.109474 can be obtained, which is only better than 3040.994404 obtained by GA.

5.5. Case 5: three-bar truss design

Stress constraints are concerned to deal with the design of a three-bar truss structure in which the volume is to be minimized. Several algorithms have been proposed to solve this problem, including Ray and Liew,[63] Zhang et al.,[42] Wang et al.,[22] Wang and Li,[23] and Zhao et al.[30]

Table 7 lists the best results from which it can be seen that ADCQGA can obtain the best objective function value. It is worth noting that several other methods also achieved the best

Table 5 Comparison of the best results of Case 3.

Algorithm	x_1	x_2	x_3	x_4	$f(\boldsymbol{x})$
Ref.[14]	0.20598600	3.47132800	9.02022400	0.20648000	1.728226
Ref.[60]	0.20570000	3.47050000	9.03660000	0.20570000	1.724852
Ref.[12]	0.20313700	3.54299800	9.03349800	0.20617900	1.733461
Ref.[64]	0.19974200	3.61206000	9.03750000	0.20608200	1.737300
Ref.[23]	0.20572964	3.47048867	9.03662391	0.20572964	1.724852
GA	0.23067867	9.56455705	3.46682033	0.40687727	2.161399
DCQGA	0.14834335	7.52063257	6.77838363	0.22055816	1.730715
ADCQGA	0.23867056	6.20184382	4.33897317	0.24928755	1.441541

Table 6 Comparison of the best results of Case 4.

Algorithm	x_1	x_2	x_3	x_4	x_5	x_6	x_7	$f(\boldsymbol{x})$
Ref.[63]	3.500068100	0.700000010	17.00000000	7.327602050	7.715321750	3.350267020	5.286654500	2994.744241
Ref.[65]	3.500010000	0.700000000	17.00000000	7.300156000	7.800027000	3.350221000	5.286685000	2996.356689
Ref.[42]	3.500000000	0.700000000	17.00000000	0.730000000	7.7153199115	3.3502146610	5.286654465	2994.471066
Ref.[23]	3.500000000	0.700000000	17.00000000	7.300000000	7.7153199115	3.3502146661	5.286654465	2994.471066
GA	3.518113390	0.702207490	17.00114714	7.590601700	7.888346340	3.376082080	5.311879240	3040.994404
DCQGA	3.511620780	0.700059040	17.00679272	7.302784120	7.770320230	3.354754370	5.287030010	3003.109474
ADCQGA	3.500014180	0.700000840	17.00000118	7.300093690	7.714366510	3.343933090	5.285586630	2992.184537

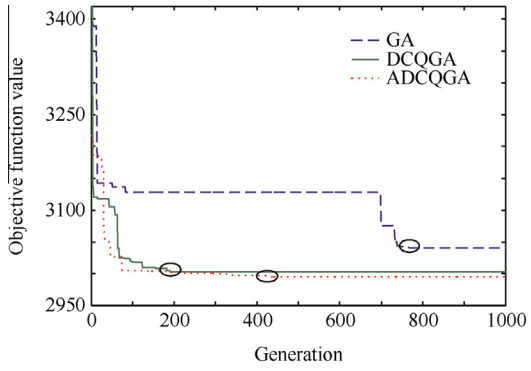


Fig. 10 Best convergence processes for Case 4.

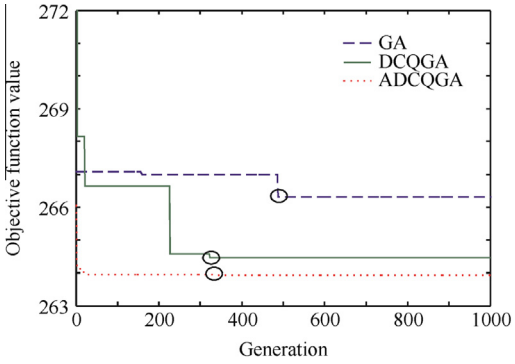


Fig. 11 Best convergence processes for Case 5.

objective function value (Zhang, Wang, Wang and Li, and Zhao). However, the best objective function value obtained by DCQGA is 264.4663393, larger than that of ADCQGA. Fig. 11 shows the best convergence processes of GA, DCQGA and ADCQGA in 50 independent runs.

From the results of the five well-known constrained engineering design problems discussed above, it can be seen that ADCQGA achieved the best objective function value compared with the existing algorithms included in this paper. Not only the search capability of ADCQGA is better than DCQGA, but also it performs well in terms of convergence rate.

6. Application of ADCQGA to constrained multi-aircraft cooperative target allocation problem

6.1. Problems description

The air combat and optimization problems have attracted the interest of several people from different disciplines.[66–68] In this paper, we use the model proposed in Ref.[45] Assume N red fighters are approaching, and M blue fighters are assigned to intercept. Each blue fighter carries L missiles and the total missile number is $Z = ML$. And all missiles should be assigned with a target. The aim is to minimize the evaluation function[45], which is defined in:

$$E(\pi) = \sum_{n=1}^{N}\sum_{m=1}^{M}\left[\mathrm{th}_{nm}\left(\prod_{k=1}^{Z}(1-\mathrm{th}_{kn})^{X_{kn}}\right)\right]$$

$$\text{s.t.}\begin{cases}\sum_{n=1}^{N} X_{kn} = 1, & k = 1,2,\cdots,Z\\ \sum_{k=1}^{Z} X_{kn} \geqslant 1, & n = 1,2,\cdots,N\\ \sum_{k=1}^{Z} X_{kn} \leqslant 2, & n = 1,2,\cdots,N\end{cases} \tag{19}$$

Table 7 Comparison of the best results of Case 5.

Algorithm	x_1	x_2	$f(\boldsymbol{x})$
Ref.[63]	0.7886210370	0.4084013340	263.8958466
Ref.[42]	0.7886751359	0.4082482868	263.8958434
Ref.[22]	0.3567292035	0.0516895376	263.8958434
Ref.[23]	0.7886751287	0.4082483070	263.8958434
Ref.[30]	0.7886751338	0.4082482928	263.8958434
GA	0.3840133272	0.8058071829	266.3180221
DCQGA	0.7958603229	0.3936304683	264.4663393
ADCQGA	0.7886751360	0.4082482872	263.8958434

where $n = 1, 2, \cdots, N$; $m = 1, 2, \cdots, M$; $k = 1, 2, \cdots, Z$; th_{nm} is the threat of the nth red fighter to the mth blue fighter, th_{kn} the threat of the blue fighter that carried missile k to the red fighter. The value of X_{kn} is 1 or 0; $X_{kn} = 1$ indicates that missile k is assigned to the nth red fighter. Additionally, the constraints are described in Eq. (19).

In this paper, $B_m(m = 1, 2, \cdots, M)$ denotes the mth blue fighter and $R_n(n = 1, 2, \cdots, N)$ denotes the nth red fighter. As in Ref.[45], the threat of R_n to B_m can be described as[45]

$$\text{th}_{nm} = \omega_1 \text{th}_{nm}^{D_{nm}} \text{th}_{nm}^{\varepsilon_{nm}} + \omega_2 \text{th}_{nm}^{V_{R_n}} \tag{20}$$

where

$$\text{th}_{nm}^{D_{nm}} = \begin{cases} 1, & D_{nm} \leqslant D_{\text{Missile}} \\ 1 - \dfrac{D_{nm} - D_{\text{Missile}}}{D_{\text{Radar}} - D_{\text{Missile}}}, & D_{\text{Missile}} < D_{nm} \leqslant D_{\text{Radar}} \\ 0, & D_{nm} > D_{\text{Radar}} \end{cases} \tag{21}$$

$$\text{th}_{nm}^{\varepsilon_{nm}} = \text{e}^{-\lambda_1 (\pi \varepsilon_{nm}/180)^{\lambda_2}} \tag{22}$$

$$\text{th}_{nm}^{V_{R_n}} = \begin{cases} 1, & V_{B_m} < 0.5 V_{R_n} \\ 1.5 - \dfrac{V_{B_m}}{V_{R_n}}, & 0.5 V_{R_n} \leqslant V_{B_m} \leqslant 1.4 V_{R_n} \\ 0.1, & V_{B_m} > 1.4 V_{R_n} \end{cases} \tag{23}$$

$$\omega_1 + \omega_2 = 1 \tag{24}$$

where $\omega_1, \omega_2, \lambda_1$ and λ_2 are positive constants; $\text{th}_{nm}^{D_{nm}}$ is the distance threat factor, $\text{th}_{nm}^{\varepsilon_{nm}}$ the bore of sight (BOS) angle threat factor, and $\text{th}_{nm}^{V_{R_n}}$ the velocity threat factor; D_{nm} denotes the distance between R_n and B_m; D_{Missile} and D_{Radar} are the missile effective range of red fighter and red radar maximum track range respectively; V_{R_n} and V_{B_m} represent the velocity of R_n and B_m; ε_{nm} is the BOS angle of B_m to R_n.

The threat th_{kn} of the blue fighter carrying missile k to the red fighter is defined in the similar way.

After obtaining th_{kn} and th_{nm}, we can calculate the evaluation function value through Eq. (19). The ultimate goal is to minimize the evaluation function value.

Before applying ADCQGA to solve the constrained multi-aircraft cooperative target allocation problem, the solution space mapping needs improving. In this paper, we replace Eq. (7) with Eq. (25).

$$\begin{cases} x_{ic} = \text{ceil}(x_i^{\text{l}} + \dfrac{\cos r_i + 1}{2}(x_i^{\text{u}} - x_i^{\text{l}})) \\ x_{is} = \text{ceil}(x_i^{\text{l}} + \dfrac{\sin r_i + 1}{2}(x_i^{\text{u}} - x_i^{\text{l}})) \end{cases} \tag{25}$$

where $x_i(i = 1, 2, \cdots, N)$ is the ith independent variable. $x_i^{\text{l}} = 0, x_i^{\text{u}} = N$ and the sign $\text{ceil}(x)$ rounds the elements to the nearest integers greater than or equal to x. Therefore, all independent variables would be an integer in $[1, N]$, which indicates the number of the assigned red fighter.

6.2. Experimental results

In a scenario, $M = 5, N = 16, L = 4$ and $Z = 20$. The velocity of all the red fighters is 300 m/s and 350 m/s for all blue fighters. The missile effective range of all blue fighters is 70 km, and the radar maximum track range of blue fighters is 120 km. The missile effective range and radar maximum track range of red fighters are 47 km and 80 km respectively. All parameters for obtaining th_{nm} and th_{kn} are listed in Table 8. After obtaining th_{nm} and th_{kn}, we can calculate the evaluation value according to Eq. (19).

As mentioned in Ref.[45], at a certain time instant, all targets are in the attackable regions of the blue fighters. Fig. 12 illustrates the situation in detail. Blue fighters fly head-on with the red fighters at the same altitude, and blue fighters decide to make a cooperative attack on the red fighters.

To assess the effectiveness and efficiency of the ADCQGA, its performance was compared against DCQGA and GA. Table 9 lists the best results and from Table 9 it can be seen that the best objective function value obtained by ADCQGA is 0.576354. However, the best objective function value obtained by DCQGA and GA are 0.599230 and 0.611542, both worse than that of ADCQGA. The best assignments of the three algorithms are shown in Tables 10–12.

Fig. 13 shows the best convergence processes of GA, DCQGA and ADCQGA in 50 independent runs. ADCQGA achieves 0.576354 in less than 40 generations. DCQGA achieves 0.599230 about in 50 generations, but the result does not change in 200 generations. For GA, it can only obtain 0.611542 in 200 generations. In conclusion, ADCQGA

Table 8 Parameters for calculating th_{nm} and th_{kn}.

Parameter	Value
Velocity of blue fighter (m/s)	350
Velocity of red fighter (m/s)	300
Missile effective range of blue fighter (km)	70
Radar maximum track range of blue fighter (km)	120
Missile effective range of red fighter (km)	47
Radar maximum track range of red fighter (km)	80
Constant λ_1	3
Constant λ_2	2
Constant ω_1	0.8
Constant ω_2	0.2

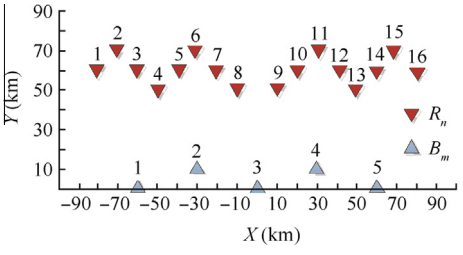


Fig. 12 Air combat situation.

Table 9 Comparison of the best results using different algorithms.

Algorithm	Best objective function value
GA	0.611542
DCQGA	0.599230
ADCQGA	0.576354

Table 10 Best assignment using GA.

Blue fighter	Missile	Assigned red fighter
B1	1 2 3 4	R1 R4 R2 R3
B2	5 6 7 8	R5 R1 R7 R6
B3	9 10 11 12	R5 R9 R7 R8
B4	13 14 15 16	R16 R10 R11 R12
B5	17 18 19 20	R14 R15 R16 R13

Table 11 Best assignment using DCQGA.

Blue fighter	Missile	Assigned red fighter
B1	1 2 3 4	R4 R1 R3 R2
B2	5 6 7 8	R6 R4 R7 R5
B3	9 10 11 12	R10 R9 R8 R8
B4	13 14 15 16	R11 R12 R16 R10
B5	17 18 19 20	R14 R16 R15 R13

Table 12 Best assignment using ADCQGA.

Blue fighter	Missile	Assigned red fighter
B1	1 2 3 4	R3 R2 R1 R4
B2	5 6 7 8	R6 R5 R7 R4
B3	9 10 11 12	R9 R5 R8 R8
B4	13 14 15 16	R10 R12 R11 R18
B5	17 18 19 20	R16 R13 R15 R14

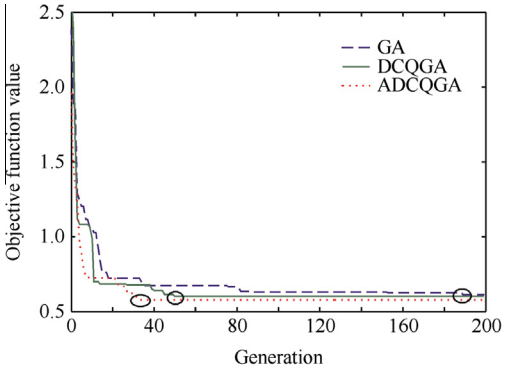


Fig. 13 Best convergence processes for constrained multi-aircraft cooperative target allocation.

performs better than DAQGA and GA for constrained multi-aircraft cooperative target allocation.

6.3. Discussions of ADCQGA for constrained optimization problems

In summary, there are several explanations for the good performance of ADCQGA: (A) all solutions including feasible and infeasible solutions were used as indicators to obtain the best solution; (B) double-individuals were implemented in ADCQGA, and the two chains of each double-individual underwent Q-gate rotation and mutation simultaneously, on the one hand, it makes ADCQGA more efficient; on the other hand, the search region is expanded since each double-individual consists of a pair of individuals; (C) the AEP and replacement technique improve the efficiency of ADCQGA; (D) adaptive mutation was implemented to enrich the population diversity. Using these improved techniques, the ADCQGA achieves better performance in terms of effectiveness and efficiency compared with the existing algorithms mentioned in this literature.

7. Conclusions

(1) This paper presents ADCQGA for solving constrained optimization problems.
(2) ADCQGA utilizes adaptive rotation, adaptive evolution process, mutation and replacement technique to improve the search efficiency and convergence rate.
(3) The effectiveness and efficiency of ADCQGA are demonstrated by its results on thirteen benchmark functions and five well-known constrained engineering design problems.
(4) ADCQGA is successfully applied to solving a constrained multi-aircraft cooperative target allocation problem.

Acknowledgments

The work was supported by the National Natural Science Foundation of China (No. 61004089). It has also been supported by China Scholarship Council.

References

1. Michalewicz Z, Schoenauer M. Evolutionary algorithms for constrained parameter optimization problems. *Evol Comput* 1996;**4**(1):1–32.
2. Eiben A. *Theoretical aspects of evolutionary computing*. Berlin: Springer; 2001. p. 13–30.
3. Coello Coello CA. Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. *Computer Methods Appl Mech Eng* 2002;**191**(11):1245–87.
4. Salcedo-Sanz S. A survey of repair methods used as constraint handling techniques in evolutionary algorithms. *Computer Sci Rev* 2009;**3**(3):175–92.
5. Courant R. Variational methods for the solution of problems of equilibrium and vibrations. *Bull Am Math Soc* 1943;**49**(1):23.
6. Deb K. An efficient constraint handling method for genetic algorithms. *Computer Methods Appl Mech Eng* 2000;**186**(2):311–38.
7. Coello Coello CA. Use of a self-adaptive penalty approach for engineering optimization problems. *Comput Ind* 2000;**41**(2):113–27.
8. Nanakorn P, Meesomklin K. An adaptive penalty function in genetic algorithms for structural design optimization. *Comput Struct* 2001;**79**(29):2527–39.
9. Barbosa HJC, Lemonge ACC. A new adaptive penalty scheme for genetic algorithms. *Info Sci* 2003;**156**(3):215–51.
10. Farmani R, Wright JA. Self-adaptive fitness formulation for constrained optimization. *IEEE Trans Evol Comput* 2003;**7**(5):445–55.
11. Wu WH, Lin CY. The second generation of self-organizing adaptive penalty strategy for constrained genetic search. *Adv Eng Software* 2004;**35**(12):815–25.

12. Huang FZ, Wang L, He Q. An effective co-evolutionary differential evolution for constrained optimization. *Appl Math Comput* 2007;**186**(1):340–56.
13. Puzzi S, Carpinteri A. A double-multiplicative dynamic penalty approach for constrained evolutionary optimization. *Struct Multidiscip Optim* 2008;**35**(5):431–45.
14. Coello Coello CA, Mezura Montes E. Constraint-handling in genetic algorithms through the use of dominance-based tournament selection. *Adv Eng Inform* 2002;**16**(3):193–203.
15. Orvosh D, Davis L. Shall we repair genetic algorithms, combinatorial optimization and feasibility constraints. *Proceedings of the fifth international conference on genetic algorithms*; 1993. p. 650.
16. Orvosh D, Davis L. Using a genetic algorithm to optimize problems with feasibility constraints. *IEEE conference on evolutionary computation*; 1994. p. 548–53.
17. Chootinan P, Chen A. Constraint handling in genetic algorithms using a gradient-based repair method. *Comput Oper Res* 2006;**33**(8):2263–81.
18. Smith AE, Coit DW. Constraint handling techniques-penalty functions*Handbook of evolutionary computation*. Oxford: Oxford University Press and Institute of Physics Publishing; 1997. p. 1–6.
19. Bean JC. Genetic algorithms and random keys for sequencing and optimization. *ORSA J Comput* 1994;**6**(2):154–60.
20. Kowalczyk R. Constraint consistent genetic algorithms. *IEEE international conference on evolutionary computation*; 1997. p. 343–8.
21. Takahama T, Sakai S. Constrained optimization by the ε constrained differential evolution with gradient-based mutation and feasible elites. *2006 IEEE congress on evolutionary computation*; 2006. p. 1–8.
22. Wang Y, Cai Z, Zhou Y, Fan Z. Constrained optimization based on hybrid evolutionary algorithm and adaptive constraint-handling technique. *Struct Multidiscip O* 2009;**37**(4):395–413.
23. Wang L, Li LP. An effective differential evolution with level comparison for constrained engineering design. *Struct Multidiscip O* 2010;**41**(6):947–63.
24. Paredis J. *Co-evolutionary constraint satisfaction. Parallel problem solving from nature-PPSN III*. Berlin: Springer; 1994. p. 46–55.
25. Powell D, Skolnick MM. Using genetic algorithms in engineering design optimization with non-linear constraints. *Proceedings of the 5th international conference on genetic algorithms*; 1993. p. 424–31.
26. Venter G, Haftka R. Constrained particle swarm optimization using a bi-objective formulation. *Struct Multidiscip Optim* 2010;**40**(1–6):65–76.
27. Adeli H, Cheng NT. Augmented Lagrangian genetic algorithm for structural optimization. *J Aerosp Eng* 1994;**7**(1):104–18.
28. Fung RY, Tang J, Wang D. Extension of a hybrid genetic algorithm for nonlinear programming problems with equality and inequality constraints. *Comput Oper Res* 2002;**29**(3):261–74.
29. Chen PH, Shahandashti SM. Hybrid of genetic algorithm and simulated annealing for multiple project scheduling with multiple resource constraints. *Autom Constr* 2009;**18**(4):434–43.
30. Zhao JQ, Wang L, Zeng P, Fan WH. An effective hybrid genetic algorithm with flexible allowance technique for constrained engineering design optimization. *Expert Syst Appl* 2012;**39**(5): 6041–51.
31. Shor PW. Algorithms for quantum computation: discrete logarithms and factoring. *Proceedings of the 35th annual symposium on foundations of computer science*; 1994. p. 124–34.
32. Grover LK. A fast quantum mechanical algorithm for database search. *Proceedings of the 28th annual ACM symposium on theory of computing*; 1996. p. 212–9.
33. Narayanan A, Moore M. Quantum-inspired genetic algorithms. *Proceedings of IEEE international conference on evolutionary computation*; 1996. p. 61–6.
34. Han KH, Park KH, Lee CH, Kim JH. Parallel quantum-inspired genetic algorithm for combinatorial optimization problem. *Proceedings of the 2001 congress on evolutionary computation*; 2001. p. 1422–9.
35. Han KH, Kim JH. Quantum-inspired evolutionary algorithm for a class of combinatorial optimization. *IEEE Trans Evol Comput* 2002;**6**(6):580–93.
36. Yang JA, Li B, Zhuang Z. Multi-universe parallel quantum genetic algorithm its application to blind-source separation. *Proceedings of the 2003 international conference on neural networks and signal*; 2003. p. 393–8.
37. Chen H, Zhang JS, Zhang C. Chaos updating rotated gates quantum-inspired genetic algorithm. *2004 international conference on communications, circuits and systems*; 2004. p. 1108–12.
38. Li Y, Zhang Y, Zhao R, Jiao L. The immune quantum-inspired evolutionary algorithm. *2004 IEEE international conference on systems, man and cybernetics*; 2004. p. 3301–5.
39. Yang S, Wang M, Jiao L. A novel quantum evolutionary algorithm and its application. *2004 CEC congress on evolutionary computation*; 2004. p. 820–6.
40. Wang L, Tang F, Wu H. Hybrid genetic algorithm based on quantum computing for numerical optimization and parameter estimation. *Appl Math Comput* 2005;**171**(2):1141–56.
41. Zhang XX, Henan PR. Quantum-inspired immune evolutionary algorithm. *2008 ISBIM'08 international seminar on business and information management*; 2008. p. 323–5.
42. Zhang M, Luo W, Wang X. Differential evolution with dynamic stochastic selection for constrained optimization. *Info Sci* 2008;**178**(15):3043–74.
43. Li SY, Li PC. Quantum genetic algorithm based on real encoding and gradient information of object function. *J Harbin Inst Technol* 2008;**38**(8):1216–8 [Chinese].
44. Li S, Li P. *Quantum computation and quantum optimization algorithms*. Harbin: Harbin Institute of Technology Press; 2009 [Chinese].
45. Luo DL, Shen CL, Wang B, Wu WH. Air combat decision-making for cooperative multiple target attack using heuristic adaptive genetic algorithm. *Proceedings of 2005 international conference on machine learning and cybernetics*; 2005. p. 473–8.
46. Paddon HG. Maneuvering target simulation for testing the terminal guidance of air-to-air missiles. 1977. Air Force Interim Report. Report No.: ADAO39757.
47. Li N, Yi W, Gong G. Multi-aircraft cooperative target allocation in BVR air combat using cultural-genetic algorithm. *Syst Simul Sci Comput* 2012;414–22.
48. Qiang Z, Humin L, Fei W, Feiyao D. Research of genetic-ant colony fusion algorithm for multi-missile cooperative combat target assignment problem. *2009 Chinese control and decision conference (CCDC 2009)*; 2009. p. 5910–2.
49. Luo D, Duan H, Wu S, Li M. Research on air combat decision-making for cooperative multiple target attack using heuristic ant colony algorithm. *Acta Aeronautica et Astronautica Sinica* 2006;**27**(6):1166–70 [Chinese].
50. Guo H, Xu HJ, Gu XD, Liu DY. Air combat decision-making for cooperative multiple target attack based on improved particle swarm algorithm. *Fire Control Command Control* 2011;**36**(6):49–51 [Chinese].
51. Niu LW, Gao XG, Zhang K, Li B. Making decisions on proper cooperation tactics for multiple fighters to combat from beyond visual range (BVR) to within visual range (WVR). *J Northwestern Polytechnical Univ* 2011;**29**(6):971–7 [Chinese].
52. Zhang BC, Peng C, Yu XR, Zhou XH, Zou LJ. Cooperative target assignment algorithm of multiple missiles based on ant colony optimization. *J Projectiles, Rockets, Missiles Guidance* 2012;**32**(4):69–73 [Chinese].
53. Chen X, Wei X. Method of firepower allocation in multi-UCAV cooperative combat for multi-target attacking. *2012 15th international symposium on computational intelligence and design (ISCID)*; 2012.p.452-5.

54. Dasgupta S, Papadimitriou C, Vazirani U. *Algorithms*. 1st ed. New York: The Mcgraw Hill Company; 2006. p. 295–311.
55. Wang JH, Yin ZY. A ranking selection-based particle swarm optimizer for engineering design optimization problems. *Struct Multidiscip Optim* 2008;**37**(2):131–47.
56. Runarsson TP, Yao X. Stochastic ranking for constrained evolutionary optimization. *IEEE Trans Evol Comput* 2000;**4**(3):284–94.
57. Runarsson TP, Yao X. Search biases in constrained evolutionary optimization. *IEEE Trans Syst, Man, and Cybern, Part C: Appl Rev* 2005;**35**(2):233–43.
58. Mezura-Montes E, Coello CAC. A simple multimembered evolution strategy to solve constrained optimization problems. *IEEE Trans Evol Comput* 2005;**9**(1):1–17.
59. Deb K. An introduction to genetic algorithms. *Sadhana* 1999;**24**(4–5):293–315.
60. Coello Coello CA, Becerra RL. Efficient evolutionary optimization through the use of a cultural algorithm. *Eng Optim* 2004;**36**(2):219–36.
61. Ragsdell K, Phillips D. Optimal design of a class of welded structures using geometric programming. *J Manuf Sci Eng* 1976;**98**(3):1021–5.
62. Deb K. Optimal design of a welded beam via genetic algorithms. *AIAA J* 1991;**29**(11):2013–5.
63. Ray T, Liew KM. Society and civilization: An optimization algorithm based on the simulation of social behavior. *IEEE Trans Evol Comput* 2003;**7**(4):386–96.
64. Mezura-Montes E, Coello CAC. An empirical study about the usefulness of evolution strategies to solve constrained optimization problems. *Int J Gen Syst* 2008;**37**(4):443–73.
65. Mezura-Montes E, Coello CAC, Vela'zquez-Reyes J. Increasing successful offspring and diversity in differential evolution for engineering design. *Proceedings of the 7th international conference on adaptive computing in design and manufacture (ACDM 2006)*; 2006. p. 131–9.
66. Hu TY, Yu XQ. Aerodynamic/stealthy/structural multidisciplinary design optimization of unmanned combat air vehicle. *Chin J Aeronaut* 2009;**22**(4):380–6.
67. Zhang Y, Chen J, Shen LC. Real-time trajectory planning for UCAV air-to-surface attack using inverse dynamics optimization method and receding horizon control. *Chin J Aeronaut* 2013;**26**(4):1038–56.
68. Zhang X, Chen J, Xin B, Peng ZH. A memetic algorithm for path planning of curvature-constrained UAVs performing surveillance of multiple ground targets. *Chin J Aeronaut* 2014;**27**(3):622–33.

Kong Haipeng is a Ph.D. student at School of Automation Science and Electrical Engineering, Beihang University, Beijing, China. He received his B.S. degree from Qingdao University in 2010. His area of research is multi-agent based modeling and simulation technology.

Li Ni is an associate professor and master supervisor at School of Automation Science and Electrical Engineering, Beihang University, Beijing, China. She received her Ph.D. degree from the same university in 2006. Her current research interests are virtual prototyping technology, artificial simulation technology.

Shen Yuzhong is an associate professor at Department of Modeling, Simulation, and Visualization Engineering and Department of Electrical and Computer Engineering, Old Dominion University. He received his B.S. degree in electrical engineering from Fudan University, Shanghai, China, M.S. degree in computer engineering from Mississippi State University, Starkville, Mississippi, and Ph.D. degree in electrical engineering from the University of Delaware, Newark, Delaware. His research interests include signal and image processing, visualization and computer graphics, and modeling and simulation. He is also affiliated with Virginia Modeling, Analysis, and Simulation Center at Old Dominion University.