Mixed-Integer Constrained Optimization Based on Memetic Algorithm

Y. C. Lin

Department of Electrical Engineering
WuFeng University
Chiayi County, Taiwan, R.O.C.
Chien-lin@wfu.edu.tw

ABSTRACT
Evolutionary algorithms (EAs) are population-based global search methods. They have been successfully applied to many complex optimization problems. However, EAs are frequently incapable of finding a convergence solution in default of local search mechanisms. Memetic Algorithms (MAs) are hybrid EAs that combine genetic operators with local search methods. With global exploration and local exploitation in search space, MAs are capable of obtaining more high-quality solutions. On the other hand, mixed-integer hybrid differential evolution (MIHDE), as an EA-based search algorithm, has been successfully applied to many mixed-integer optimization problems. In this paper, a memetic algorithm based on MIHDE is developed for solving mixed-integer optimization problems. However, most of real-world mixed-integer optimization problems frequently consist of equality and/or inequality constraints. In order to effectively handle constraints, an evolutionary Lagrange method based on memetic algorithm is developed to solve the mixed-integer constrained optimization problems. The proposed algorithm is implemented and tested on two benchmark mixed-integer constrained optimization problems. Experimental results show that the proposed algorithm can find better optimal solutions compared with some other search algorithms. Therefore, it implies that the proposed memetic algorithm is a good approach to mixed-integer optimization problems.

Keywords: Evolutionary algorithm, memetic algorithm, mixed-integer hybrid differential evolution, Lagrange method.

1. Introduction

Evolutionary algorithms (EAs) are powerful search algorithms based on the mechanism of natural selection [1, 2]. Unlike conventional search approaches, they simultaneously consider many points in the search space so as to increase the chance of global convergence Therefore, EAs exhibit a good potential of global exploration. In the past decade, EAs have been successfully applied to many complex continuous optimization problems such as highly nonlinear, non-differentiable and multi-modal optimization problems [1-6]. However, EAs are frequently incapable of finding a precise solution in default of local search mechanisms.

In recent years, memetic algorithms (MAs) have been receiving increasing attention from the evolutionary computation community. MAs are inspired by Darwinian's principles of natural evolution and Dawkins' notion of a memes [7]. MAs are hybrid EAs that combine genetic operators with local search methods. The local search methods are used to perform the local refinement procedures [8]. Therefore, they can be regarded as the integration between population-based EAs and local search methods. With global exploration and local exploitation in search space, MAs are capable of obtaining more high-quality solutions. MAs have shown to be promising for solving optimization problems in a wide range of applications [8-11].

In real world, many optimization problems frequently contain integer variables in addition to continuous variables. For example, the standard formats of steel thickness must be chosen as integers in mechanical design. This kind of optimization problems involved continuous and integer variables are generally called mixed-integer optimization problems. Mixed-Integer Hybrid Differential Evolution (MIHDE) [12] is a mixed-integer EA-based search algorithm. A mixed coding is introduced in MIHDE to implement the evolutionary process of continuous and integer variables. In order to enhance the global exploration, the migration is devised to maintain the population diversity in MIHDE. The MIHDE has been successfully applied to many complex mixed-integer optimization problems [12-14].

In this paper, a memetic algoritm based on MIHDE, called Memetic MIHDE, is developed to deal with mixed-integer optimization problems. The Memetic MIHDE incorporates a local search method called Nelder-Mead simplex method [15] into MIHDE to enhance the local exploitation. The Nelder-Mead method is one of the most popular derivative-free nonlinear optimization methods. Instead of using the derivative information of the function to be minimized, through reflection, expansion, contraction and shrinkage search steps for a simplex, the Nelder-Mead method can maintain a non-degenerate simplex to find a local optimum.

In addition, these mixed-integer optimization problems are often coupled with nonlinear constraints so that they are more intractable because the feasible solution space is greatly suppressed by the constraints. Several methods based on EAs have been developed to solve mixed-integer optimization problems [16–18]. However, these methods are not well posed for various mixed-integer problems due to lack of a good constraint-handling mechanism. Hence, in order to tackle mixed-integer constrained optimization problems effectively, it is worthwhile to develop an efficient evolutionary mixed-integer constrained optimization method. In this paper, we can combine the Memetic MIHDE algorithm with Lagrange method to construct an evolutionary Lagrange method to deal with mixed-integer constrained optimization problems. Finally, the proposed algorithm is implemented and tested on two benchmark mixed-integer optimization problems [19]. Experimental results show that the proposed algorithm can find better optimal solutions compared with the other search algorithms. This demonstrates that the proposed memetic algorithm is a good approach to mixed-integer optimization problems.

The remainder of this paper is organized as follows. Section 2 presents an overview of Memetic MIHDE algorithm. In Section 3, The formulation of evolutionary Lagrange method based on Memetic MIHDE is developed. Computational results for mixed-integer constrained optimization problems are provided in Section 4. Finally, conclusions are drawn in Section 5.

2. Memetic MIHDE algorithm

First, consider a mixed-integer optimization problem as follows:

$$\begin{aligned} &\min_{\mathbf{x},\mathbf{y}} f(\mathbf{x},\mathbf{y}) \\ &\mathbf{x}^L \le \mathbf{x} \le \mathbf{x}^U \\ &\mathbf{y}^L \le \mathbf{y} \le \mathbf{y}^U \end{aligned} \tag{1}$$

where $\mathbf{x}$ represents an n_C-dimensional vector of real-valued variables, $\mathbf{y}$ is an n_I-dimensional vector of integer-valued variables, and $(\mathbf{x}^L,\mathbf{y}^L)$ and $(\mathbf{x}^U,\mathbf{y}^U)$ are the lower and upper bounds of the corresponding decision vectors.

Table 1 shows the basic operations of conventional evolutionary algorithms and Memetic MIHDE. Each operation of Memetic MIHDE is discussed as follows:

Evolutionary Algorithms	Memetic MIHDE
1. Initialization 2. Mutation 3. Crossover 4. Evaluation and Selection 5. Repeat step 2 to 4 until stop criterion is satisfied	1. Initialization 2. Mutation 3. Crossover 4. Evaluation and Selection 5. Migration if necessary 6. Local Search 7. Repeat step 2 to 6 until stop criterion is satisfied

Table 1. Basic operations of evolutionary algorithmsand Memetic MIHDE.

1) *Initialization*

The Memetic MIHDE algorithm uses N_p decision vectors $\{\mathbf{z}_i^G\} = \{(\mathbf{x}^G,\mathbf{y}^G)_i\}$, $i = 1,\ldots,N_p$ to denote a population of N_p individuals in the *G*-th generation. The decision vector (chromosome), $(\mathbf{x},\mathbf{y})_i$, is represented as $(x_{1i},\ldots,x_{ji},\ldots,x_{n_C i}, y_{1i},\ldots,y_{ji},\ldots,y_{n_I i})$. The decision variables (genes), x_{ji} and y_{ji}, are directly coded as real-valued and integer-valued numbers. The initialization process generates N_p

decision vectors $(\mathbf{x},\mathbf{y})_i$ randomly, and should try to cover the entire search space uniformly as in the form:

$$(\mathbf{x}^0,\mathbf{y}^0)_i = (\mathbf{x}^L,\mathbf{y}^L) + \lfloor \rho_i\{(\mathbf{x}^U,\mathbf{y}^U)-(\mathbf{x}^L,\mathbf{y}^L)\}\rfloor\ i=1,\ldots,N_p \tag{2}$$

where $\rho_i = \mathrm{Diag}(\rho_{i,1},\ \rho_{i,2},\ \ldots,\ \rho_{i,n_C+n_I})$ is a diagonal matrix, the diagonal elements $(\rho_{i,1},\ \rho_{i,2},\ \ldots,\ \rho_{i,n_C+n_I})$ are random numbers in the range $[0,1]$, the other elements are zero, and the rounding operator $\lfloor \mathbf{a}=\rho_i(\mathbf{x}^U-\mathbf{x}^L), \mathbf{b}=\rho_i(\mathbf{y}^U-\mathbf{y}^L)\}\rfloor$ in (2) is defined as $(\mathbf{a},\mathrm{INT}[\mathbf{b}])$ in which the operator $\mathrm{INT}[\mathbf{b}]$ is expressed as the nearest integer-valued vector to the real-valued vector $\mathbf{b}$.

2) Mutation

The *i*-th mutant individual $(\mathbf{u}^G,\mathbf{v}^G)_i$ is obtained by the difference for two random individuals as expressed in the form: The *i*-th mutant individual $(\mathbf{u}^G,\mathbf{v}^G)_i$ is obtained by the difference for two random individuals as expressed in the form:

$$\begin{aligned}(\mathbf{u}^G,\mathbf{v}^G)_i &= (\mathbf{x}^G,\mathbf{y}^G)_p + \lfloor \rho_m\{(\mathbf{x}^G,\mathbf{y}^G)_k-(\mathbf{x}^G,\mathbf{y}^G)_l\}\rfloor \\ &= (\mathbf{x}^G,\mathbf{y}^G)_p + (\rho_m(\mathbf{x}_k^G-\mathbf{x}_l^G), \mathrm{INT}[\rho_m(\mathbf{y}_k^G-\mathbf{y}_l^G)])\end{aligned} \tag{3}$$

where random indices $k,l \in 1,\ldots,N_p$ are mutually different. The operator $\mathrm{INT}[\mathbf{b}=\rho_m(\mathbf{y}_k^G-\mathbf{y}_l^G)]$ in (3) is to find the nearest integer vector to the real vector $\mathbf{b}$. The mutation factor ρ_m is a real random number between zero and one. This factor is used to control the search step among the direction of the differential variation $(\mathbf{x}^G,\mathbf{y}^G)_k-(\mathbf{x}^G,\mathbf{y}^G)_l$.

3) Crossover

In crossover operation, each gene of the *i*-th individual is reproduced from the mutant vector $(\mathbf{u}^G,\mathbf{v}^G)_i=(u_{1i}^G,u_{2i}^G,\ldots,u_{n_Ci}^G,v_{1i}^G,v_{2i}^G,\ldots,v_{n_Ii}^G)$ and the individual $(\mathbf{x}^G,\mathbf{y}^G)_i=(x_{1i}^G,x_{2i}^G,\ldots,x_{n_Ci}^G,y_{1i}^G,y_{2i}^G,\ldots,y_{n_Ii}^G)$ as follows:

$$u_{li}^{G+1}=\begin{cases}x_{li}^G, & \text{if a random number} > \rho_c\\ u_{li}^G, & \text{otherwise}; l=1,\ldots,n_C,\ i=1,\ldots,N_p\end{cases} \tag{4}$$

$$v_{li}^{G+1}=\begin{cases}y_{li}^G, & \text{if a random number} > \rho_c\\ v_{li}^G, & \text{otherwise}; l=1,\ldots,n_I,\ i=1,\ldots,N_p\end{cases} \tag{5}$$

where the crossover factor $\rho_c \in [0,1]$ is a constant and the value can be specified by the user.

4) Evaluation and selection

The operation includes two evaluation phases. The first phase is performed to produce the new population in the next generation as (6). The second phase is used to obtain the best individual as (7).

$$(\mathbf{x}^{G+1},y^{G+1})_i = \operatorname{argmin}\{f((\mathbf{x}^G,\mathbf{y}^G)_i), f((\mathbf{u}^{G+1},\mathbf{v}^{G+1})_i)\}\quad \text{for } i=1,\ldots,N_p \tag{6}$$

$$(\mathbf{x}^{G+1},\mathbf{y}^{G+1})_b = \operatorname{argmin}\{f((\mathbf{x}^{G+1},\mathbf{y}^{G+1})_i),\ i=1,\ldots,N_p\} \tag{7}$$

where $(\mathbf{x}^{G+1},\mathbf{y}^{G+1})_b$ is the best individual with the smallest objective function value.

5) Migration

In order to increase the exploration of the search space, a migration operation is introduced to generate a diversified population. Based on the best individual $(\mathbf{x}^{G+1},\mathbf{y}^{G+1})_b=(x_{1b}^{G+1},x_{2b}^{G+1},\ldots,x_{n_Cb}^{G+1},y_{1b}^{G+1}, y_{2b}^{G+1},\ldots,y_{n_Ib}^{G+1})$, the *j*-th gene of the *i*-th individual can be diversified by the following equations:

$$x_{ji}^{G+1}=\begin{cases}x_{jb}^{G+1}+\rho_1(x_j^L-x_{jb}^{G+1}), & \text{if a random number} < \dfrac{x_{jb}^{G+1}-x_j^L}{x_j^U-x_j^L}\\ x_{jb}^{G+1}+\rho_1(x_j^U-x_{jb}^{G+1}), & \text{otherwise};\ j=1,\ldots,n_C,\ i=1,\ldots,N_p\end{cases} \tag{8}$$

$$y_{ji}^{G+1} = \begin{cases} y_{jb}^{G+1} + \mathrm{INT}[\rho_2(y_j^L - y_{jb}^{G+1})], \text{ if a random number} < \dfrac{y_{jb}^{G+1} - y_j^L}{y_j^U - y_j^L} \\ y_{jb}^{G+1} + \mathrm{INT}[\rho_2(y_j^U - y_{jb}^{G+1})], \text{ otherwise; } j = 1,\dots,n_I,\ i = 1,\dots,N_p \end{cases} \tag{9}$$

where ρ_1 and ρ_2 are the random numbers in the range [0,1].

The migration operation in Memetic MIHDE is performed only if a measure for the population diversity is not satisfied, that is when most of individuals have clustered together, the migration has to be actuated to make some improvements. In this paper, we propose a measure called the population diversity degree η to check whether the migration operation should be performed. In order to define the measure, we first introduce the following gene diversity index for each real-valued gene x_{ji}^{G+1} and for each integer-valued gene y_{ki}^{G+1},

$$dx_{ji} = \begin{cases} 0, \text{ if } \left| \dfrac{x_{ji}^{G+1} - x_{jb}^{G+1}}{x_{jb}^{G+1}} \right| < \varepsilon_2;\ j = 1,\dots,n_C,\ i = 1,\dots,N_p,\ i \neq b \\ 1, \text{ otherwise} \end{cases} \tag{10}$$

$$dy_{ji} = \begin{cases} 0, \text{ if } y_{ji}^{G+1} = y_{jb}^{G+1};\ j = 1,\dots,n_I,\ i = 1,\dots,N_p,\ i \neq b \\ 1, \text{ otherwise} \end{cases} \tag{11}$$

where dx_{ji} and dy_{ji} are the gene diversity indices and $\varepsilon_2 \in [0,1]$ is a tolerance for real-valued gene provided by the user. According to (10) and (11), we assign the *j*-th gene diversity index for the *i*-th individual to zero if this gene clusters to the best gene. We now define the population diversity degree η as a ratio of total diversified genes in the population. From (10) and (11) we have the population diversity degree as

$$\eta = \left\{ \sum_{\substack{i=1 \\ i \neq b}}^{N_p} \left[\sum_{j=1}^{n_C} dx_{ji} + \sum_{j=1}^{n_I} d_{ji} \right] \right\} / \left\{ (n_C + n_I)(N_p - 1) \right\} \tag{12}$$

From equations (10), (11) and (12), the value of η is in the range [0, 1]. Consequently, we can set a tolerance for population diversity, $\varepsilon_1 \in [0,1]$, to assess whether the migration should be actuated. If η is smaller than ε_1, then Memetic MIHDE performs the migration to generate a new population to escape a local solution. Contrary, if η is not less than ε_1, then Memetic MIHDE suspends the migration operation to keep a constant search direction to a target solution.

6) *Local search*

The local search procedure of Nelder-Mead method [15] is used after migration operation at each generation. The Nelder-Mead method is a nonlinear programming approach in continuous domain. In mixed-integer programming, a simplex is devised to search for the optimal solution with respect to continuous decision variables by fixing the values of integer decision variables of each candidate solution. Therefore, the local search procedure refines the candidate solution to approach a local optimum. As a result, The Memetic MIHDE algorithm can enhance the global exploration and local exploitation for the mixed-integer optimization problems.

3. Evolutionary Lagrange method based on Memetic MIHDE algorithm

In order to solve mixed-integer constrained optimization problems, a Lagrange method based on memetic MIHDE is developed to handle the constraints. First, let us consider a mixed-integer constrained optimization problem as follows:

$$\begin{aligned} &\min_{\mathbf{x},\mathbf{y}} f(\mathbf{x},\mathbf{y}) \\ \text{subject to } \ & h_j(\mathbf{x},\mathbf{y}) = 0,\ j = 1,\dots,m_e \\ & g_j(\mathbf{x},\mathbf{y}) \leq 0,\ j = 1,\dots,m_i \end{aligned} \tag{13}$$

where $\mathbf{x}$ represents an n_C-dimensional vector of continuous variables, $\mathbf{y}$ is a n_I-dimensional vector of integer variables, and $h_j(\mathbf{x},\mathbf{y})$ and $g_j(\mathbf{x},\mathbf{y})$ stand for the equality and inequality constraints respectively. To abbreviate these expressions, a compact notation $\mathbf{z} = (\mathbf{x},\mathbf{y})$ is used in the following discussions, and the problem is referred to as primal problem.

An augmented Lagrange function corresponding to the primal problem is defined by:

$$L_a(\mathbf{z},\nu,\upsilon) = f(\mathbf{z}) + \sum_{k=1}^{m_e} \alpha_k \left\{ [h_k(\mathbf{z}) + \nu_k]^2 - \nu_k^2 \right\} + \sum_{k=1}^{m_i} \beta_k \left\{ \langle g_k(\mathbf{z}) + \upsilon_k \rangle_+^2 - \upsilon_k^2 \right\} \quad (14)$$

where α_k and β_k are positive penalty parameters, the bracket operation is denoted as $\langle g \rangle_+ = \max\{g, 0\}$, and $\nu = (\nu_1, ..., \nu_{m_e})$ and $\upsilon = (\upsilon_1, ..., \upsilon_{m_i}) \geq \mathbf{0}$ are the corresponding Lagrange multipliers.

In nonlinear programming, the saddle point theorem [20] states that, if a point is a saddle point of the augmented Lagrange function associated with the primal problem, then the point is the solution of the primal problem. Accordingly, the saddle point theorem can be used to solve mixed-integer constrained optimization problems.

If $(\mathbf{z}^*, \nu^*, \upsilon^*)$ is a saddle point of the augmented Lagrange function $L_a(\mathbf{z},\nu,\upsilon)$, then $(\mathbf{z}^*, \nu^*, \upsilon^*)$ satisfies the following condition:

$$L_a(\mathbf{z}^*,\nu,\upsilon) \leq L_a(\mathbf{z}^*,\nu^*,\upsilon^*) \leq L_a(\mathbf{z},\nu^*,\upsilon^*) \quad (15)$$

The saddle point can be obtained by minimizing $L_a(\mathbf{z},\nu^*,\upsilon^*)$ with the optimal Lagrange multipliers (ν^*,υ^*) as a fixed parameter vector. However, the difficulty of this minimization is that it requires the knowledge of (ν^*,υ^*) previously. In general, the optimal values of Lagrange multipliers are unknown *a priori*. The duality theorem can be employed to overcome this difficulty.

According to the duality theorem, the primal problem (13) can be transformed into a dual problem as follows:

$$\max_{(\nu,\upsilon)} H(\nu,\upsilon) \quad (16)$$

where the dual function $H(\nu,\upsilon)$ is denoted as

$$H(\nu,\upsilon) = \min_{\mathbf{z}} L_a(\mathbf{z},\nu,\upsilon) \quad (17)$$

and the set $\Xi = \{(\nu,\upsilon) | H(\nu,\upsilon) \text{ exists, and } \upsilon \geq \mathbf{0}\}$. Based on (16) and (17), the duality theorem can be expressed by the following duality relationship statements:

1) $\mathbf{z}^* = (\mathbf{x}^*, \mathbf{y}^*)$ is a minimizer of the primal problem (13).

2) (ν^*,υ^*) with $\upsilon^* \geq \mathbf{0}$ is a maximizer of the dual problem (16).

3) $H(\nu^*,\upsilon^*) = f(\mathbf{z}^*)$. That is, the saddle point of the augmented Lagrange function (14) is the optimal solution of the primal problem (13).

According to the duality relations, we can construct an evolutionary max-min algorithm to solve mixed-integer constrained optimization problems. The evolutionary max-min algorithm includes two phases as stated in Table 2. In the first phase (step 2 in Table 2), the Memetic MIHDE is used to minimize the augmented Lagrange function with multipliers fixed. In the second phase (step 3 in Table 2), the Lagrange multipliers are updated to ascend the value of the dual function toward obtaining maximization of the dual problem.

```
Step 1. Set initial iteration index: $l = 0$.
        Set initial multipliers: $\nu_k^l = 0$ for $k = 1,...,m_e$, $\upsilon_k^l = 0$ for $k = 1,...,m_i$.
        Set penalty parameters: $\alpha_k > 0$ for $k = 1,...,m_e$, $\beta_k > 0$ for $k = 1,...,m_i$.
Step 2. Use Memetic MIHDE to minimize $L_a(\mathbf{z},\mathbf{v}^l,\boldsymbol{\upsilon}^l)$.
        Let $\mathbf{z}_b^l = (\mathbf{x}^l,\mathbf{y}^l)_b$ be a minimum solution to the function $L_a(\mathbf{z},\mathbf{v}^l,\boldsymbol{\upsilon}^l)$.
Step 3. Update the multipliers as follows:
            $\nu_k^{l+1} = h_k(\mathbf{z}_b^l) + \nu_k^l$
            $\upsilon_k^{l+1} = \langle g_k(\mathbf{z}_b^l) + \upsilon_k^l \rangle_+$
Step 4. Update $\alpha_k$ and $\beta_k$, if necessary.
Step 5. Stop if stopping criterion is satisfied.
        Otherwise, let $l = l + 1$ and repeat Steps 2 to 4.
```

Table 2. Evolutionary max-min algorithm.

As far as the evolutionary computation is concerned, the evolutionary max-min procedure may increase many function evaluations and affect the convergence rate. However, in order to find the exact solution, it is necessary and inevitable unless using other special approaches, e.g., sequential quadratic programming (SQP) [21], to continuously update the Lagrange multipliers. Unfortunately, SQP is not applicable for the non-differentiable mixed-integer constrained optimization problems. The update of the Lagrange multipliers is based on the exact or approximate minimum of the augmented Lagrange function with multipliers fixed. As presented by Arora *et al.* [21], an exact or approximate minimum is necessary in order to ensure proper shift of the Lagrange function towards the required solution. With a rough minimum, the shift of the Lagrange function may be far away from the required solution leading to obtain a nonexistent dual function so that the duality theorem shall be disobeyed. The saddle point is accordingly unable to be obtained using the rough shift. Therefore, in order to obtain an exact or approximate minimum, sufficient iterations for Memetic MIHDE minimization phase are necessary for the evolutionary max-min algorithm.

In addition, constant penalty parameters are used in the max-min algorithm. These parameters are suited for solving linear constrained problems because the contours of $L_a(\mathbf{z}, \nu, \upsilon)$ do not change shape from generation to generation. However, if the constraints are highly nonlinear, the contours of $L_a(\mathbf{z}, \nu, \upsilon)$ still highly depend on the values of penalty parameters. This fact indicates that the evolutionary max-min algorithm in Table 2 may be sensitive to the setting values of penalty parameters. Small penalty parameters, for example, might lead to an infeasible solution. On the other hand, large penalty parameters might cause the penalty functions to be ill conditioned. In order to increase the possibility of global convergence, a self-adaptation scheme for the penalty parameters is embedded into the evolutionary max-min algorithm as shown in Table 3. The evolutionary max-min algorithm with self-adaptation scheme is called Memetic MIHDE MIHDE-Adaptive MaxMin (Memetic MIHDE-AMM) algorithm. It is extended from Powell's multiplier algorithm [22] towards improving the extent of the constraint violations.

Step 1. Set initial iteration: $l = 0$.

Set initial multipliers: $\nu_k^l = 0$ for $k = 1, \dots, m_e$, $\upsilon_k^l = 0$ for $k = 1, \dots, m_i$.

Set penalty parameters: $\alpha_k > 0$ for $k = 1, \dots, m_e$, $\beta_k > 0$ for $k = 1, \dots, m_i$. Set tolerance for the maximum constraint violation, ε_K (e.g. $\varepsilon_K = 10^{32}$).

Step 2. Use Memetic MIHDE to solve $L_a(\mathbf{z}, \mathbf{v}^l, \boldsymbol{\upsilon}^l)$. Let $\mathbf{z}_b^l = (\mathbf{x}^l, \mathbf{y}^l)_b$ be a minimum solution to the function $L_a(\mathbf{z}, \mathbf{v}^l, \boldsymbol{\upsilon}^l)$.

Step 3. Evaluate the maximum constraint violation as

$$\hat{\varepsilon}_K = \max\left\{\max_k |h_k|, \max_k\left[\langle g_k + \upsilon_k \rangle_+ - \upsilon_k\right]\right\},$$

and establish the following sets for the equality and inequality constraints whose violation is not improved by the factor ω_1:

$$I_E = \left\{k : |h_k| > \varepsilon_K / \omega_1, k = 1, \dots, m_e\right\}$$

$$I_I = \left\{k : \left|\left[\langle g_k + \upsilon_k \rangle_+ - \upsilon_k\right]\right| > \varepsilon_K / \omega_1, k = 1, \dots, m_i\right\}$$

Step 4. If $\hat{\varepsilon}_K \geq \varepsilon_K$, let $\alpha_k = \omega_2 \alpha_k$, $\nu_k^{l+1} = \nu_k^l / \omega_2$ for all $k \in I_E$, and $\beta_k = \omega_2 \beta_k$, $\upsilon_k^{l+1} = \upsilon_k^l / \omega_2$ for all $k \in I_I$. Go to Step 7.

Otherwise, go to Step 5.

Step 5. Update the multipliers as follows:

$$\nu_k^{l+1} = h_k(\mathbf{z}_b^l) + \nu_k^l$$
$$\upsilon_k^{l+1} = \left\langle g_k(\mathbf{z}_b^l) + \upsilon_k^l \right\rangle_+$$

Step 6. If $\hat{\varepsilon}_K \leq \varepsilon_K / \omega_1$, let $\varepsilon_K = \hat{\varepsilon}_K$ and go to Step 7.

Otherwise, let $\alpha_k = \omega_2 \alpha_k$, $\nu_k^{l+1} = \nu_k^{l+1} / \omega_2$ for all $k \in I_E$, $\beta_k = \omega_2 \beta_k$, $\upsilon_k^{l+1} = \upsilon_k^{l+1} / \omega_2$ for all $k \in I_I$, and $\varepsilon_K = \hat{\varepsilon}_K$. Go to Step 7.

Step 7. Stop if stopping criterion is satisfied. Otherwise, let $l = l + 1$ and repeat Steps 2 to 6.

Table 3. Evolutionary max-min algorithm with adaptive penalty parameters.

In Memetic MIHDE-AMM algorithm, a scalar factor ω_1 is used to enforce an improvement for the constraint violations. The other scalar factor ω_2 is used to increase the values of penalty parameters. For implementation, we used $\omega_1 = 4$ and $\omega_2 = 10$ as suggested in Powell's algorithm [22]. Owing to using the self-adaptation scheme, the initial penalty parameter values do not affect the final search result except for the convergence rate. This implies that the algorithm is insensitive to the penalty parameters. Steps 3, 4 and 6 in Table 3 are used to improve the constraint violation and update the penalty parameters. In step 4, if the maximum constraint violation does not improve (i.e. $\hat{\varepsilon}_K \geq \varepsilon_K$), for those unimproved constraints (i.e. $k \in I_E \cup I_I$) the associated penalty parameters can be increased by the factor ω_2 and the associated multipliers can be decreased by the same factor. In step 5, if the maximum constraint violation improves (i.e. $\hat{\varepsilon}_K < \varepsilon_K$), the penalty parameters keep unchanged and the multipliers are updated. In step 6, if the maximum constraint violation does not be improved by the factor ω_1, for those unimproved constraints the associated penalty parameters can be increased by the factor ω_2 and the associated multipliers can be decreased by the same factor. And then the new maximum constraint violation $\hat{\varepsilon}_K$ replaces the previous maximum constraint violation ε_K. Because the penalty parameters are automatically updated, the maximum constraint violation $\hat{\varepsilon}_K$ will keep descending until all of the constraints are satisfied so that finally the Memetic MIHDE-AMM can converge to the saddle point of the augmented Lagrange function.

4. Experimental examples

Two typical mixed-integer optimization problem presented by Sandgren [19] are used to test the Memetic MIHDE and Memetic MIHDE-AMM algorithm. For implementation, the setting parameters used in Memetic MIHDE are listed as follows: population size $N_P = 5$, crossover factor $\rho_c = 0.5$, and two tolerances $\varepsilon_1 = \varepsilon_2 = 0.1$. In order to illustrate their performance of global search, the other three algorithms, IDCNLP, SA and MVEP [18, 23, 24], are chosen for comparison.

Problem 1:

The mechanical optimization problem is an optimal design problem of compound gear train as shown in Figure 1. It is desired to produce a gear ratio as possible to 1/6.931. For each gear, the number of teeth must be between 12 and 60. The design variables are the numbers of teeth, which must be integers.

$$\mathbf{y} = (T_d, T_b, T_a, T_f) = (y_1, y_2, y_3, y_4)$$

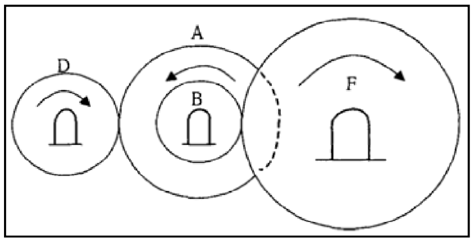

Figure 1. Gear train design.

The optimization problem is formulated as:

$$\min_{\mathbf{y}} f(\mathbf{y}) = \left(\frac{1}{6.931} - \frac{T_d T_b}{T_a T_f}\right)^2 = \left(\frac{1}{6.931} - \frac{y_1 y_2}{y_3 y_4}\right)^2 \qquad (18)$$

subject to $12 \leq y_i \leq 60,\ i = 1, \ldots, 4$.

Computational results are shown in Table 4. The objective value obtained by Memetic MIHDE is lowest. This indicates that the optimal solution obtained by Memetic MIHDE is better than those obtained by IDCNLP, SA and MVEP. Therefore, the Memetic MIHDE algorithm is a good approach to global optimization.

Item	IDCNLP	SA	MVEP	Memetic MIHDE
y_1	14	30	30	19
y_2	29	15	15	16
y_3	47	52	52	43
y_4	59	60	60	49
$f(\mathbf{y})$	4.5×10^{-6}	2.36×10^{-9}	2.36×10^{-9}	2.70×10^{-12}

Table 4. Computational results with different solvingalgorithms.

Problem 2:

The mechanical optimization problem is an optimal design problem of pressure vessel as shown in Figure 2. The design variables are the dimensions required for the specifications of the vessel, i.e. $(\mathbf{x}, \mathbf{y}) = (x_1, x_2, y_1, y_2)$.

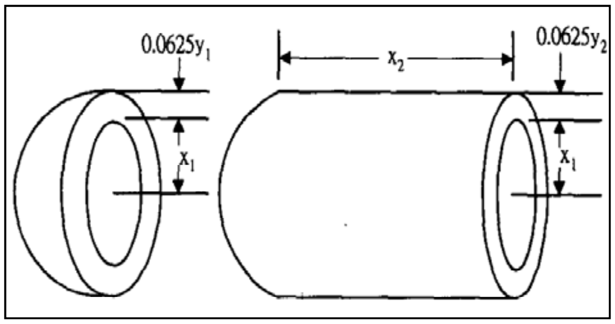

Figure 2. Pressure vessel design.

The objective function is the combined costs of material, forming and welding of the pressure vessel. The constraints are set in accordance with the respective ASME codes. The mixed-integer constrained optimization problem is expressed as:

$$\min_{\mathbf{x},\mathbf{y}} f(\mathbf{x},\mathbf{y}) = 0.6224(0.0625y_1)x_1x_2 + 1.7781(0.0625y_2)x_1^2 + 3.1661(0.0625y_1)^2 x_2 + 19.84(0.0625y_1)^2 x_1 \quad (19)$$

subject to

$$g_1(\mathbf{x},\mathbf{y}) = 0.0193x_1 - 0.0625y_1 \le 0$$
$$g_2(\mathbf{x},\mathbf{y}) = 0.00954x_1 - 0.0625y_2 \le 0$$
$$g_3(\mathbf{x},\mathbf{y}) = 750\times1728 - \pi x_1^2 x_2 - {}^4\!/_3\,\pi x_1^3 \le 0$$
$$g_4(\mathbf{x},\mathbf{y}) = x_2 - 240 \le 0$$
$$10.0 \le x_1 \le 100.0$$
$$10.0 \le x_2 \le 240.0$$
$$10 \le y_1 \le 32$$
$$10 \le y_2 \le 32$$

Computational results are shown in Table 5. The objective value obtained by Memetic MIHDE-AMM is lowest and all constraints are satisfied ($g_i \le 0$). This demonstrates that the optimal solution obtained by Memetic MIHDE-AMM is better than those obtained by IDCNLP, SA and MVEP. Therefore, the proposed Memetic MIHDE-AMM algorithm is appropriate for solving mixed-integer constrained optimization problems.

5. Conclusions

In this paper, a memetic algorithm, Memetic MIHDE, is developed to solve mixed-integer optimization problems. Combined Memetic MIHDE algorithm with Lagrange method, an evolutionary Lagrange method, Memetic MIHDE-AMM algorithm, can be implemented to deal with mixed-integer constrained optimization problems. Finally, the proposed method is applied to two typical mixed-integer optimization problems. Computational results show the proposed method is superior to the other three solving algorithms in searching global solution. This demonstrates that the proposed evolutionary Lagrange method based on Memetic MIHDE algorithm can effectively solve mixed-integer constrained optimization problems. Therefore, it implies that the proposed memetic algorithm is a good approach to mixed-integer optimization problems.

Item	IDCNLP	SA	MVEP	Memetic MIHDE-AMM
x_1	48.3807	58.2900	51.1958	38.8571
x_2	111.7449	43.6930	90.7821	221.4116
y_1	18	18	16	12
y_2	10	10	10	10
g_1	-0.1913	-0.0250	-0.0119	-0.0001
g_2	-0.1634	-0.0689	-0.1366	-0.2543
g_3	-75.8750	-6.5496	-13584.5631	-1.9921
g_4	-128.2551	-196.3070	-149.2179	-18.5884
f	8048.6190	7197.7	7108.6160	6521.9778

Table 5. Computational results with different solving algorithms.

References

[1] Z. Michalewicz, “Genetic Algorithm + Data Structure = Evolution Programs”, Springer-Verlag, 1994.

[2] T. Back, D. Fogel and Z. Michalewicz, “Handbook of Evolutionary Computation”, New York: Oxford Univ. Press, 1997.

[3] A. Afkar, M. Mahmoodi-Kaleibar and A. Paykani, “Geometry optimization of double wishbone suspension system via genetic algorithm for handling improvement”, *Journal of Vibroengineering*, vol. 14, pp. 827–837, 2012.

[4] M. J. Richard, M. Bouazara, L. Khadir and G. Q. Cai, “Structural optimization algorithm for vehicle suspensions”, *Trans. Can. Soc. Mech. Eng.*, vol. 35, pp. 1–17, 2011.

[5] F. Yaman and A. E. Yılmaz, “Impacts of genetic algorithm parameters on the solution performance for the

uniform circular antenna array pattern synthesis problem", *Journal of Applied Research and Technology*, vol. 8, no. 3, pp. 378–394, 2010.

[6] A. Vargas-Martínez and L. E. Garza-Castanon, "Combining artificial intelligence and advanced techniques in fault-tolerant control", *Journal of Applied Research and Technology*, vol. 9, no. 2, pp. 202–226, 2011.

[7] R. Dawkins, "The Selfish Gene", Oxford Univ. Press, 1976.

[8] W. E. Hart, N. Krasnogor and J. E. Smith, "Recent Advances in Memetic Algorithms", Springer-Verlag, 2005.

[9] A. Quintero and S. Pierre, "A memetic algorithm for assigning cells to switches in cellular mobile networks", *IEEE Commun. Lett.*, vol. 6, no. 11, pp. 484–486, 2002.

[10] H. Ishibuchi, T. Yoshida and T. Murata, "Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling", *IEEE Trans. Evol. Comput.*, vol. 7, no. 2, pp. 204–223, 2003.

[11] M. Tang and X. Yao, "A memetic algorithm for VLSI floorplanning", *IEEE Trans. Syst., ManCybern. B, Cybern.* , vol. 37, no. 1, pp. 62–69, 2007.

[12] Y. C. Lin, K. S. Hwang and F. S. Wang, "A mixed-coding scheme of evolutionary algorithms to solve mixed-integer nonlinear programming problems", *Computers and Mathematics with Applications*, vol. 47, pp. 1295-1307, 2004.

[13] Y. C. Lin, Y. C. Lin and K. L. Su, "Production planning based on evolutionary mixed-integer nonlinear programming", *ICIC Express Letters*, vol. 4, no. 5(B), pp. 1881-1886, 2010.

[14] Y. C. Lin, Y. C. Lin, K. L. Su and W. C. Chang, "Identification of control systems using evolutionary neural networks", *ICIC Express Letters*, vol. 5, no. 4(B), pp. 1307-1312, 2011.

[15] J. A. Nelder and R. Mead, "A simplex method for function minimization", *Computer J.*, vol. 7, no. 4, pp. 308-313, 1965.

[16] T. Yokota, M. Gen and Y. X. Li, "Genetic algorithm for non-linear mixed integer programming problems and its applications", *Computers & Industrial Engineering J.*, vol. 30, pp. 905-917, 1996.

[17] B. K. S. Cheung, A. Langevin and H. Delmaire, "Coupling genetic algorithm with a grid search method to solve mixed integer nonlinear programming problems", *Comput. Math. Applic.*, vol. 34, pp. 13-23, 1997.

[18] Y. J. Cao and Q. H. Wu, "Mechanical design optimization by mixed-variable evolutionary programming", in *Proc. IEEE Int. Conf. Evolutionary Computation*, *Indianapolis*, 1997, pp. 443-446.

[19] E. Sandgren, "Nonlinear integer and discrete programming in mechanical design", *ASME J. Mechanical Design*, vol. 112, pp. 223-229, 1990.

[20] D. A. Wismer and R. Chattergy, "Introduction to Nonlinear Optimization", Elsevier North-Holland, 1978.

[21] J. S. Arora, A. I. Chahande and J. K. Paeng, "Multiplier methods for engineering optimization", *Int. J. Numerical Methods in Engineering*, vol.32, pp.1485-1525, 1991.

[22] M. J. D. Powell, "Algorithms for nonlinear constraints that use Lagrangian functions", *Math. Programming*, vol. 14, pp. 224-248, 1978.

[23] J. F. Fu, R. G. Fenton and W. L. Cleghorn, "A mixed integer-discrete-continuous programming method and its application to engineering design optimization", *Engineering Optimization*, vol.17, no. 3, pp.236-280, 1991.

[24] C. Zhang and H. P. Wang, "Mixed-discrete nonlinear optimization with simulated annealing", *Engineering Optimization*, vol. 21, pp.277-291, 1993.