COPYRIGHT NOTICE

FedUni ResearchOnline
<http://researchonline.ballarat.edu.au>

This is the submitted for peer-review version of the following article:

Long, Q., & Wu, C. (2014). A hybrid method combining algorithm and Hooke-Jeeves method for constrained global optimization. *Journal of Industrial and Management Optimization*, 10(4). 1279-1296

Which has been published in final form at:

<http://dx.doi.org/10.3934/jimo.2014.10.1279>

© American Institute of Mathematical Sciences.

This is the author's version of the work. It is posted here with permission of the publisher for your personal use. No further distribution is permitted.

A HYBRID METHOD COMBINING GENETIC ALGORITHM AND HOOKE-JEEVES METHOD FOR CONSTRAINED GLOBAL OPTIMIZATION

QIANG LONG

School of Science, Information, Technology and Engineering
University of Ballarat, Mt Helen, 3350, Victoria, Australia

CHANGZHI WU

School of Built Environment,
Curtin University, Perth 4845, WA, Australia

(Communicated by the associate editor name)

ABSTRACT. A new global optimization method combining genetic algorithm and Hooke-Jeeves method to solve a class of constrained optimization problems is studied in this paper. We first introduce the quadratic penalty function method and the exact penalty function method to transform the original constrained optimization problem with general equality and inequality constraints into a sequence of optimization problems only with box constraints. Then, the combination of genetic algorithm and Hooke-Jeeves method is applied to solve the transformed optimization problems. Since Hooke-Jeeves method is good at local search, our proposed method dramatically improves the accuracy and convergence rate of genetic algorithm. In view of the derivative-free of Hooke-Jeeves method, our method only requires information of objective function value which not only can overcome the computational difficulties caused by the ill-condition of the square penalty function, but also can handle the non-differentiability by the exact penalty function. Some well-known test problems are investigated. The numerical results show that our proposed method is efficient and robust.

1. Introduction. In this paper, we consider the following constrained optimization problem

$$(\text{COP}) \quad \begin{cases} \text{Minimize} & f(x) \\ \text{Subject to} & g_i(x) \leq 0 \quad i = 1, \dots, m, \\ & h_j(x) = 0 \quad j = 1, \dots, l, \\ & x \in X, \end{cases} \quad (1)$$

where $f : \mathbb{R}^n \rightarrow \mathbb{R}$, $g_i : \mathbb{R}^n \rightarrow \mathbb{R}$, $i = 1, \dots, m$ and $h_j : \mathbb{R}^n \rightarrow \mathbb{R}$, $j = 1, \dots, l$ are Lipschitz continuous functions; $X \subset \mathbb{R}^n$ is a box set, i.e.,

$$X = \{x = (x_1, x_2, \dots, x_n) \mid l_i \leq x_i \leq u_i, \quad i = 1, \dots, n\}.$$

2010 *Mathematics Subject Classification.* Primary: 90c46, 90c90, Secondary: 90c26.

Key words and phrases. Hybrid method, Genetic algorithm, Hooke-Jeeves method, Penalty function method, Constrained global optimization.

The second author is partially supported by NSFC 11001288, the Key Project from CME 210179, the natural science foundations of Chongqing: cstc2013jjB0149 and cstc2013jcyjA1338.

It is worth to note that our requirement for $f, g_1, \dots, g_m, h_1, \dots, h_l$ is just Lipschitz continuity, not necessary differentiability. Thus, our method is particularly suitable for the case in which the gradients of the objective function and the constrained functions are not available or hard to obtain ([1, 9, 17, 18, 25]).

Constraint handling technique is one of the major concerns to solve constrained optimization problems. A number of constraint-handling techniques have been developed to solve COP [8, 20, 23, 31]. In [29], these techniques are classified as three categories: i) methods based on penalty functions, ii) methods based on biasing feasible over infeasible solutions, and iii) methods based on multi-objective optimization concepts. Among them, the penalty-function-based methods are the most popular due to its simplicity in theory and ease in implementation. The key idea of this method is by appending the violations of constraints into the objective function so that the original constrained optimization problems are transformed into unconstrained ones by using an auxiliary function which is known as penalty function. Generally speaking, there are two different classes of penalty function methods, i.e., *interior penalty function method* [6, 13] and *exterior penalty function method* [5, 21, 22].

Exterior penalty function method is to generate a sequence of infeasible points such that they have a limit. Furthermore, this limit is an optimal solution to the original problem. A special exterior penalty function method is *exact penalty function method* [11, 26, 30]. The advantage of this method is that with a reasonably large penalty parameter which is not required to approach infinity, it can yield an optimum solution to the original problem through solving the transformed unconstrained problems. However, since the maximum function and absolute function are involved, the transformed unconstrained optimization problems are nonsmooth by this method. Thus, all the smooth-based optimization methods [2] cannot be applied. To overcome this drawback, derivative-free-based method is a good option.

In recent years, there is a trend to develop a kind of global optimization methods by combining stochastic methods and deterministic methods. Stochastic methods, such as genetic algorithm, simulated annealing and particle swarm method, are known as their global search but defective at local search. However, deterministic methods, such as Newton's method, conjugate gradient method and coordinate search method, are efficient at local search. The investigation of how to combine the two methods has attracted many research interests. So far a lot of works have contributed to this idea, such as [7] where tabu search and Nelder-Mead simplex method are combined, [10] where Nelder-Mead simplex method and genetic algorithm are combined, and [15] where simulated annealing and direct search method are combined.

In this paper, we will develop a new hybrid method in which a derivative-free method, say Hooke-Jeeves method [2], and a stochastic method, say genetic algorithm [12, 28] are combined together to solve COP. The key idea is that during the implementation of genetic algorithm, except crossover operator, mutation operator and selection operator, we incorporate another operator, called as acceleration operator which is designed from Hooke-Jeeves method. The acceleration is achieved through applying Hooke-Jeeves method to some selected chromosomes. Thus, some genes are marked as outstanding ones in the new generation.

The rest of this paper are arranged as follows. In section 2, we review exterior penalty function method and propose two transformed models of constrained global optimization. Their advantages and disadvantages are analyzed. In section 3, we

propose a new hybrid method, abbreviated as GAHJ, to solve constrained global optimization problems. In section 4, some test problems are investigated and the results are compared with available constrained optimization solvers. Section 5 concludes this paper.

2. Preliminaries. Penalty function methods transform a constrained optimization problem into a sequence of unconstrained optimization problems. The constraints are appended to the objective function via a penalty parameter and a penalty function. In general, a feasible penalty function should admit a positive penalty for infeasible points and no penalty for feasible points. Taking Problem COP for example, for the inequality constraints $g_i(x) \leq 0$, $i = 1, \dots, m$ and equality constraints $h_j(x) = 0$, $j = 1, \dots, l$, a feasible penalty function should be in the form of

$$\alpha(x) = \sum_{i=1}^m \phi[g_i(x)] + \sum_{j=1}^l \psi[h_j(x)], \quad (2)$$

where ϕ and ψ are continuous functions satisfying the following

$$\phi(y) = 0 \quad \text{if } y \leq 0 \quad \text{and} \quad \phi(y) > 0 \quad \text{if } y > 0; \quad (3)$$

$$\psi(y) = 0 \quad \text{if } y = 0 \quad \text{and} \quad \psi(y) > 0 \quad \text{if } y \neq 0. \quad (4)$$

Typically, ϕ and ψ are of the form

$$\begin{aligned} \phi(y) &= [\max\{0, y\}]^p, \\ \psi(y) &= |y|^p, \end{aligned} \quad (5)$$

where p is a positive integer. For such a case, the penalty function α can be rewritten as

$$\alpha_p(x) = \sum_{i=1}^m [\max\{0, g_i(x)\}]^p + \sum_{j=1}^l |h_j(x)|^p. \quad (6)$$

Let $g = (g_1, g_2, \dots, g_m)^T$ and $h = (h_1, h_2, \dots, h_l)^T$ be vector functions on $\mathbb{R}^n$ and $X = [l, u]$ be a box set. Then, COP can be rewritten in a vector form which is referred to as *Primal Problem*,

$$\begin{cases} \text{Minimize} & f(x) \\ \text{Subject to} & g(x) \leq 0 \\ & h(x) = 0 \\ & x \in X, \end{cases} \quad (7)$$

Using the penalty function supplied in (2), a *Penalty Problem* corresponding to (7) can be stated as

$$\begin{cases} \text{Maximize} & \theta(\mu) \\ \text{Subject to} & \mu \geq 0, \end{cases} \quad (8)$$

where $\theta(\mu) = \inf\{f(x) + \mu\alpha(x) \mid x \in X\}$ and μ is the penalty parameter. An important relationship between the primal problem and penalty problem is [2]

$$\inf\{f(x) \mid x \in X, g(x) \leq 0, h(x) = 0\} = \sup_{\mu \geq 0} \theta(\mu) = \lim_{\mu \rightarrow \infty} \theta(\mu).$$

From this relationship, it is clear that we can get arbitrarily close to the optimal objective function value of the primal problem by solving $\theta(\mu)$ for a sufficiently large μ .

Set $p = 2$ in (6). We obtain a quadratic penalty function

$$\alpha_2(x) = \sum_{i=1}^m [\max\{0, g_i(x)\}]^2 + \sum_{j=1}^l |h_j(x)|^2,$$

and the corresponding *auxiliary problem* is

$$\text{(Model 1): } \begin{cases} \text{Minimize} & f(x) + \mu\alpha_2(x) \\ \text{Subject to} & x \in X. \end{cases} \quad (9)$$

For each penalty parameter μ_k , let x_k^* be an optimal solution of (9). Then, x_k^* can be considered as an approximate solution of the primal problem. From the relationship between the primal problem and the penalty problem, we know that $x_k^* \rightarrow x^*$ when $\mu_k \rightarrow \infty$, where x^* is a solution of the primal problem. Thus, the better approximate solution of the primal problem, the larger penalty parameter μ_k is required. However, a large parameter μ_k makes (9) encounter the so-called ill-condition which may cause serious computational difficulties [2].

In numerical point of view, initial point plays a key role in solving problem (9), especially when penalty parameter μ is large. Most algorithms use penalty functions with a sequence of increasing penalty parameters. During the penalty parameter updating, the approximate optimal solution of Problem (9) with the old parameter is taken as an initial point for solving Problem (9) with the new parameter.

To avoid penalty parameter μ to be infinity, exact penalty function method, which is a special exterior penalty function method, was developed. In (6), let $p = 1$, then

$$\alpha_1(x) = \sum_{i=1}^m [\max\{0, g_i(x)\}] + \sum_{j=1}^l |h_j(x)|$$

and the corresponding *auxiliary problem* is

$$\text{(Model 2): } \begin{cases} \text{Minimize} & f(x) + \mu\alpha_1(x) \\ \text{Subject to} & x \in X, \end{cases} \quad (10)$$

It can be proved that under certain regular assumptions, when μ exceeds a threshold, the solution of the auxiliary problem (10) is exactly as the solution of the primal problem [30]. The advantage of exact penalty function is that the penalty parameter does not require to be infinite. However, maximum function and absolute function cause the corresponding auxiliary optimization problem (10) to be nonsmooth. Thus, all the gradient-based methods cannot be applied. To overcome this difficulty, we develop a derivative-free hybrid method, entitled as GAHJ, which combines genetic algorithm and Hooke-Jeeves method to solve problem (9) and (10).

3. Hybrid method: GAHJ. GAHJ is a hybrid method combining genetic algorithm and Hooke-Jeeves method. In order to improve the local search of genetic algorithm, an acceleration operator based on Hooke-Jeeves method is embedded into genetic algorithm. During the implementation of genetic algorithm, the acceleration operator generates some outstanding chromosomes, which dramatically promote the convergence rate and accuracy of optimal solution. In the following, we first introduce initial population generator, arithmetic crossover operator, nonuniform mutation operator, acceleration operator and selection operator. Then, the pseudocode of GAHJ is presented.

3.1. Initial population generator. Note that $X = [l, u]$ is the box constraint in problem (7), where $l, u \in \mathbb{R}^n$ are vectors. During the implementation of genetic algorithm, the initial population is randomly generated from X , and the number of chromosomes in the initial population equals population size. The process for generating initial population is illustrated as follows.

Initial population generator

Step 1: Input population size: *popu_size*, upper bound u and lower bound l , respectively. Set $k = 1$.

Step 2: if $k \leq \text{popu_size}$, then generate the k^{th} initial chromosome by

$$x_i = l_i + \alpha_i(u_i - l_i), \quad i = 1, 2, \dots, n,$$

where α_i , $i = 1, 2, \dots, n$ are randomly chosen number in $[0, 1]$.

Step 3: Set $k = k + 1$ and go back to Step2.

3.2. Arithmetic crossover operator. For the crossover operator, we use arithmetic crossover. Suppose that x_1 and x_2 are two chromosomes randomly selected to crossover, then the following rule is used to generate their offsprings

$$\begin{aligned} x'_1 &= \beta x_1 + (1 - \beta)x_2 \\ x'_2 &= \beta x_2 + (1 - \beta)x_1, \end{aligned} \tag{11}$$

where $\beta \in [0, 1]$ is a random number. The process of arithmetic crossover operator is given as follows.

Arithmetic crossover operator

Step 1: Input crossover rate: *cross_rate*, and population size: *popu_size*. Let counter $k = 1$.

Step 2: When $k \leq \text{popu_size}$, generate a number $\alpha \in [0, 1]$, if $\alpha \leq \text{cross_rate}$, then the k^{th} chromosome is marked as a candidate to crossover; otherwise, set $k = k + 1$ and go back to Step 2

Step 3: Sequentially choose two chromosomes which were marked as candidates for crossover, and crossover them using the strategy (11). The chromosomes obtained are stored as offspring.

Chromosomes generated by arithmetic crossover are actually convex combinations of x_1 and x_2 . This crossover operator, on one hand, sufficiently searches the local area, on the other hand, guarantees some global exploration. Additionally, this strategy is simple, direct and easy to implement. The drawback of this crossover is that only those points between x_1 and x_2 are considered, which reduces search area of crossover operator. To overcome it, we enlarge the random number β from $[0, 1]$ to $[-1, 1]$.

3.3. Nonuniform mutation operator. Nonuniform mutation is applied in mutation operator. For a given parent x , if its component x_k (here, subscript represents the k th element of vector x) was chosen to mutate, then the offspring should be

$$x' = (x_1, \dots, x'_k, \dots, x_n).$$

Here, x'_k is randomly chosen from the following two options

$$\begin{aligned} & x_k = x_k + d(t, x_k^U - x_k) \quad \text{if } \gamma \leq 0.5 \\ \text{or} & \\ & x'_k = x_k + d(t, x_k - x_k^L) \quad \text{if } \gamma > 0.5, \end{aligned} \tag{12}$$

FIGURE 1. The first two phases of Hooke-Jeeves method.

where $\gamma \in [0, 1]$ is a randomly chosen number, x_k^U and x_k^L are upper bound and lower bound of x_k , respectively. The function $d(t, y)$ is chosen to satisfy: $d(t, y) \in [0, y]$ and $\lim_{t \rightarrow \infty} d(t, y) = 0$. For example,

$$d(t, y) = yr(1 - \frac{t}{T})^b,$$

where r is a random number between 0 and 1, T is the maximal generation time, and b is the parameter for nonuniform degree which is set to be 1 in our numerical tests. The function $d(t, y)$ allows a large mutation of the selected chromosome at the earlier generations but a slight mutation when iteration achieves the maximum generation time. This trend is reasonable since at the early generations global exploration is emphasized and at the later generations local exploitation is emphasized.

The process of nonuniform mutation operator is described as follows.

Nonuniform mutation operator

Step 1: Input the mutation rate: *mutate_rate*, population size: *popu_size*, dimension of the problem: n , upper bound u , lower bound l and the maximal generation time: $T = \text{max_gene}$. Set counter $i = 1$ for chromosomes, set counter $k = 1$ for elements of each chromosome.

Step 2: For the i th chromosome (denoted as x), when $k \leq n$, generate a number $\alpha \in [0, 1]$, if $\alpha \leq \text{mutate_rate}$, then the element x_k mutates according to the strategies provided in (12); otherwise, set $k = k + 1$ and go back to Step 2.

Step 3: If $i < \text{popu_size}$, then, let $i = i + 1$ and $k = 1$, go back to Step 2; otherwise, stop the loop.

3.4. Acceleration operator. Acceleration operator is based on Hooke-Jeeves method [2] which is a derivative-free method. Hooke-Jeeves method includes two types of search: exploratory search and pattern search. The first two iterations of the procedure are illustrated in Figure 1. Given x_1 , an initial exploratory search along the coordinate directions produces a point x_2 (set as y). A pattern search along direction $x_2 - x_1$ leads to a point x_1' . Another exploratory search from x_1' gives a point x_2' (set as y'). The next pattern search is conducted along the direction $y' - y$, yielding x_1'' . The process is then repeated. An acceleration operator based on Hooke-Jeeves method is illustrated as follows.

Acceleration Operator

Step 1: Input the starting point x_0 , an initial step length t_0 and a tolerance parameter ϵ .

- Step 2:** Do initial exploratory search: starting from $x_1 (= x_0)$, run line search along the coordinate axes with the initial step length t_0 , set the point obtained as x_n and the direction $d = x_n - x_1$. Let $y = x_n$.
- Step 3:** Do pattern search: starting from y , run a line search along the direction d with an initial step size t_0 , set the obtained point as x_1 .
- Step 4:** Exploratory search: starting from x_1 , using initial step length t_0 , run a line search along the coordinate axes. Set point obtained as x_n and denote the direction $d = x_n - y$. let $y = x_n$.
- Step 5:** If $|f(y) - f(x_n)| < \epsilon$, then stop; otherwise, go back to Step 3.

Line search plays an extraordinarily important role in Hooke-Jeeves method. It is required both in pattern search and exploratory search. In general, an optimal line search is applied in Hooke-Jeeves method, but this may cause some technical issues for problems whose derivative is time consuming or impossible to achieve. Furthermore, optimal line search is time consuming itself and not global convergence. So in our simulations, we use discrete step length to simplify the computational process and avoid the computation of derivative. More precisely, a double step size strategy is introduced instead of optimal line search. At each iteration, this method starts from a small given step size. If the current step size decrease objective function value along the considered direction, we accept it and further test its double. Otherwise, we stop line search and take the last accepted step size as a solution.

The initial step size at each generation is chosen according to the following rule:

$$t_0 = \alpha \min_{x_i, x_j \in P} \|x_i - x_j\|,$$

where $\alpha \in [0, 1]$ is a parameter and P is the current generation. Clearly, the choice of initial step size is self-adaptive in this strategy. At the early stage, the diversity of population is large. So the step sizes should be bigger to guarantee enough decrease of the objective function value. At the latter stage, the population gradually converges to an approximate solution, the decrease of objective function becomes tiny vibration of individuals, which needs the step size to be smaller.

3.5. Algorithm: GAHJ. Embedding the acceleration operator to the general process of genetic algorithm, we can add some better chromosomes to the offspring, which, in return, generates more outstanding points in the next generation. However, if the acceleration operator is frequently called during the iterations, the process becomes time consuming and a lot of calculations are actually needless although it can provide more better chromosomes to the next generation. Thus, the acceleration operator should be applied as less as possible. In GAHJ, we run acceleration operator when the current generation decreases the objective function value, i.e., the best point of the current generation is better than the current best point.

For the selection operator, we keep the better chromosomes to the next generation so as to guarantee the local exploitation. On the other hand, it is still very important to maintain the diversity of the next generation which guarantees the global exploration. Therefore, Instead of keeping all the better points in the next generation or randomly choosing points to the next generation, we build it by half choosing from the best chromosomes and half choosing randomly. In the following, we present the pseudocode of the method GAHJ in which $P(t)$ and $O(t)$ stand for parents and offspring in the t^{th} generation, respectively.

Genetic algorithm with Hooke-Jeeves method (GAHJ)

1: Initialization

- 1.1: Generate the initial population $P(0)$,
- 1.2: Set crossover rate, mutation rate, and maximal generation number,
- 1.3: Set $t \leftarrow 0$.
- 2: While generation counter have not reach the maximal generation number ,
do
 - 2.1: Arithmetic crossover operator: generate $O(t)$,
 - 2.2: Nonuniform mutation operator: generate $O(t)$,
 - 2.3: Evaluate $P(t)$ and $O(t)$: compute their value of fitness function,
 - 2.4: Selection operator: build the next generation by choosing half population size of best chromosomes from $P(t)$ and $O(t)$, the other half is chosen randomly.
 - 2.5: Acceleration operator: update the best point so far, if the best objective function value decreases, then acceleration operator is activated with the updated best point as s starting point.
 - 2.6: Set $t \leftarrow t + 1$, go to 2.1
- end
- end

3.6. Parameters. The choice of parameters is important to numerical performance of genetic algorithm [8], which is the same in GAHJ. For GAHJ, because of the acceleration operator, generation time and population size can be dramatically reduced. Especially for a convex optimization problem, if acceleration operator is activated at a proper starting point, the optimal solution can be obtained in just a few generations. In our simulations, the parameters of GAHJ are various for different experimental tests. Empirically, if the dimension of the problem is n , then, the population size is $2n \sim 5n$, maximal generation number is $20n \sim 50n$. Crossover rate and mutation rate are $0.4 \sim 0.5$ and $0.1 \sim 0.2$, respectively.

The penalty parameter can be regulated through solving a sequence of auxiliary problems assigned by increasing penalty parameters. The starting point of the current auxiliary problem is chosen as the solution point of the previous auxiliary problem. This process is repeated until stop criterions are satisfied. In our method, the penalty parameter is chosen through our experiment experience. Because GAHJ does not dependent on the structure of objective function, ill-condition by exterior penalty function and nondifferentiability by exact penalty function would not exists. Therefore, we intend to choose a moderately large penalty parameter to guarantee that the solution of auxiliary problem is close to the optimal solution of the original problem.

4. Numerical experiments. In this section, we firstly show the improvement of GAHJ comparing with genetic algorithm and Hooke-Jeeves method by testing some unconstrained global optimization problems. Then, GAHJ is applied to solve some constrained global optimization problems. We firstly compare the numerical performances of two different penalty models, i.e., quadratic penalty function model (Model 1 or Problem 9) and exact penalty function model (Model 2 or Problem 10). Then, numerical comparisons of GAHJ with other constrained global optimization solvers are presented.

All test problems are solved in an environment of MATLAB(2010a) installed on an ACER ASPIRE4730Z laptop with a 2G RAM and a 2.16GB CPU.

TABLE 1. Main properties of global test problems

<i>Pro.</i>	Dim.	Multi.	Property	f^{**}	<i>Pro.</i>	Dim.	Multi.	Property	f^{**}
Ackley	10	+	Nonlinear	0	Matyas	2	—	Quadratic	0
Beale	2	+	Quadratic	0	Mich	2	+	Nonlinear	-1.8013
Bh1	2	—	Quadratic	0	Perm	4	+	Quadratic	0
Bh2	2	—	Quadratic	0	Perm0	4	+	Quadratic	0
Bh3	2	—	Quadratic	0	Powell	10	+	Quadratic	0
Booth	2	+	Quadratic	0	Powerl	4	—	Quadratic	0
Branin	2	—	Quadratic	0.397887	Rast	10	+	Nonlinear	0
Colville	4	+	Quadratic	0	Rosen	10	+	Quadratic	0
Dp	10	+	Quadratic	0	Schw	10	+	Nonlinear	0
Easom	2	+	Exponential	-1	Shekel	4	+	Nonlinear	-10.1532
Gold	2	+	Quadratic	3	Shub	2	+	Nonlinear	-186.7309
Griewank	10	+	Nonlinear	0	Sphere	10	—	Quadratic	0
Hart3	3	+	Exponential	-3.86278	Sum	10	—	Quadratic	0
Hart6	6	+	Exponential	-3.32237	Trid	10	—	Quadratic	-210
Hump	2	—	Quadratic	0	Zakh	10	—	Quadratic	0
Levy	2	+	Quadratic	0					

4.1. Improvement of GAHJ. The test problems we considered here are taken from website [14]. Main properties (like number of variables, global minimal values and number of local minimums) of those test problems are illustrated in Table 1. From Table 1, we can see that those test functions share very different properties (quadratic, nonlinear and exponential). Some of them are with multimodal functions (“+” in the column of Multi), such as Ackley, Beale, Rosenberg. The others are with unimodal functions (“—” in the column of Multi). In the last column, global minimal values (f^{**}) of the test problems are provided.

One of the most important criteria to evaluate numerical performance of a hybrid method is success rate. For a certain solver applying on a test case, success rate is defined as a ratio of successful implementation out of total implementation, i.e.,

$$\text{success rate} = \frac{\text{successful implementation}}{\text{total implementation}} \times 100\%.$$

In our numerical experiments, the total implementation for each solver on each problem is 100, and the successful implementation is identified by checking if the following certification is satisfied or not,

$$\frac{f^* - f^{**}}{|f^{**}| + 1} < \epsilon. \quad (13)$$

Here, f^* and f^{**} stand for the obtained optimal solution and the best known optimal solution, respectively, and ϵ is a threshold parameter which normally set as 10^{-2} to 10^{-3} . In our tests, we set $\epsilon = 0.01$. For a solution f^* obtained by a certain implementation, if the criteria (13) is satisfied, then, this implementation is marked as a successful one.

Table 2 shows the success rate of genetic algorithm (GA) [12], Hooke-Jeeves method (HJM) [2] and GAHJ. As illustrated in Table 2, GAHJ shares higher success rate than genetic algorithm and Hooke-Jeeves method, except for Problem Griewank, Schw and Shekel whose success rates for all methods are not good. Thus, GAHJ dramatically increased the success rate of implementations comparing with genetic algorithm and Hooke-Jeeves method.

Table 3 illustrates the mean optimal solutions and best optimal solutions obtained by genetic algorithm, Hooke-Jeeves method and GAHJ. This table clearly show that GAHJ shares better mean value of optimal solutions and better best optimal

TABLE 2. Success rate of GA, HJM and GAHJ

<i>Pro.</i>	GA	HJM	GAHJ	<i>Pro.</i>	GA	HJM	GAHJ
Ackley	100%	0	100%	Matyas	95%	100%	100%
Beale	49%	55%	99%	Mich	100%	40%	100%
Bh1	88%	16%	100%	Perm	50%	18%	76%
Bh2	52%	17%	88%	Perm0	11%	65%	80%
Bh3	27%	27%	100%	Powell	93%	100%	100%
Booth	84%	100%	100%	Powerl	14%	40%	96%
Branin	98%	100%	100%	Rast	100%	0	100%
Colville	3%	95%	100%	Rosen	50%	70%	97%
Dp	4%	80%	84%	Schw	9%	0	8%
Easom	66%	0	99%	Shekel	67%	18%	62%
Gold	93%	41%	100%	Shub	100%	3%	100%
Griewank	7%	4%	3%	Sphere	100%	100%	100%
Hart3	100%	62%	100%	Sum	100%	100%	100%
Hart6	80%	60%	80%	Trid	26%	100%	100%
Hump	100%	54%	100%	Zakh	100%	100%	100%
Levy	100%	8%	100%				

TABLE 3. Optimal solution of GA, HJM and GAHJ

Pro.	GA		HJM		GAHJ	
	Mean	Best	Mean	Best	Mean	Best
Ackley	1.7030e-3	3.6309e-4	NA	NA	7.4699e-5	1.6083e-5
Beale	2.4199e-3	4.9422e-10	3.6887e-6	1.7672e-12	5.3117e-8	2.0463e-12
Bh1	9.6912e-4	3.5799e-9	3.8914e-6	2.3670e-7	3.5173e-8	2.1649e-14
Bh2	1.3126e-3	6.5088e-8	3.2859e-6	8.5831e-8	2.6459e-8	1.5371e-10
Bh3	2.4095e-3	1.4673e-6	2.6624e-6	6.3153e-8	6.2294e-8	1.2779e-10
Booth	1.6981e-3	1.1983e-9	9.2772e-7	7.5103e-10	1.0102e-8	7.1194e-11
Branin	0.39839616	0.39788736	0.39788788	0.39788737	0.39788736	0.39788736
Colville	3.7671e-3	1.2762e-3	1.6187e-3	1.0102e-6	4.8907e-5	2.8462e-7
Dp	3.6067e-3	1.9037e-3	3.5881e-5	1.1372e-5	3.5417e-7	7.6044e-8
Easom	-0.9992	-1.000	NA	NA	-1.0000	-1.0000
Gold	3.0004e	3.0000	3.0000	3.0000	3.0000	3.0000
Griewank	3.7381e-3	2.70554e-5	1.8491e-3	1.2512e-7	2.1566e-3	6.9771e-10
Hart3	-3.8627821	-3.8627821	-3.8627735	-3.8627821	-3.8627821	-3.8627821
Hart6	-3.322368	-3.322368	-3.3223562	-3.3223665	-3.322368	-3.322368
Hump	6.1244e-6	4.7023e-8	9.0280e-7	7.7145e-8	5.1466e-8	4.6510e-8
Levy	1.3002e-6	1.2356e-12	7.9621e-8	2.2267e-9	6.1342e-10	2.7779e-13
Matyas	1.0268e-3	1.0016e-9	8.1760e-8	1.9044e-11	7.7440e-10	1.3936e-12
Mich	-1.8013015	-1.8013034	-1.8012992	-1.8013032	-1.8013034	-1.8013034
Perm	4.3802e-2	2.9972e-2	4.1476e-3	6.4493e-4	4.3802e-3	3.7047e-5
Perm0	3.0159e-3	7.0799e-5	1.7952e-3	3.4083e-5	4.3545e-4	1.7280e-6
Powell	2.6516e-3	1.2913e-5	3.5674e-4	5.7200e-5	5.7931e-6	1.9761e-8
Powerl	3.6177e-3	2.2595e-4	3.3697e-3	1.6532e-4	1.1608e-3	3.7456e-6
Rast	3.8679e-5	5.1703e-6	NA	NA	8.9499e-8	3.1739e-9
Rosen	3.2343e-2	2.0023e-2	1.2024e-3	2.6323e-4	8.3192e-6	1.5894e-6
Schw	2.0295e-4	1.3179e-4	NA	NA	1.2727e-4	1.2727e-4
Shekel	-10.536396	-10.53641	-10.536379	-10.536404	-10.53641	-10.53641
Shub	-186.71839	-186.73091	-186.73083	-186.73087	-186.73091	-186.73091
Sphere	1.9139e-7	1.6077e-8	8.3811e-7	1.5133e-7	4.2131e-10	1.4684e-11
Sum2	3.4753e-6	7.3293e-8	4.4740e-6	1.2453e-6	6.8728e-9	3.03166e-10
Trid	-204.07581	-209.72716	-210	-210	-210	-210
Zakh	9.3696e-4	4.7725e-5	1.0248e-4	1.5609e-5	1.0781e-6	6.1889e-8

solutions for all the test problems except Problem Perm. For Problem Perm, the mean value of optimal solutions is slightly less than that obtained by Hooke-Jeeves method.

Figure 2 depicts a comparison of convergence rate between GAHJ and GA on test cases Ackley, Griewank, Hart3 and Schw. For Problem Ackley and Schw, the optimal solutions are obtained by GAHJ (the solid line) at the 1300-iteration and the 133-iteration, respectively, which are much more better than those obtained by genetic algorithm (the dashed line). Furthermore, GAHJ reached the optimal

FIGURE 2. Comparison of convergence rate between GAHJ and GA.

solution just at the second iteration for Problem Griewank and Hart3, while genetic algorithm took much more iterations. This is because if the starting point is chosen properly, the acceleration operator can reach the optimal solution directly. The advantage of high accuracy of GAHJ can also be identified from these figures. For all the test cases depicted in Figure 2, solutions obtained by GAHJ are better than GA. Given the comparison made in Table 3 and Figure 2, it can be clearly observed that GAHJ can obtain optimal solutions with higher precision than genetic algorithm.

From the above discussions, we can observe that GAHJ inherited not only the global exploration of genetic algorithm, but also the local exploitation of Hooke-Jeeves method. GAHJ, as a global optimization solver, can provide optimal solutions with higher accuracy and success rate.

4.2. Solving constrained optimization problems by GAHJ. In this subsection, we evaluate the performances of GAHJ by testing thirteen well-known benchmarks. All test functions are taken from [27]. The main characteristics of these test cases are reported in Table 4. These problems include not only different types

TABLE 4. Main characteristics of 13 benchmark test functions

Pro.	Dim	Type of f	Feasible ratio	LI	NE	NI	f^{**}
Pro1	13	Quadratic	0.0003%	9	0	0	-15.0000
Pro2	20	Nonlinear	99.9962%	1	0	1	-0.803619
Pro3	10	Nonlinear	0.0000%	0	1	0	-1.0000
Pro4	5	Quadratic	26.9557%	0	0	6	-30665.539
Pro5	4	Nonlinear	0.0000%	2	3	0	5126.498
Pro6	2	Nonlinear	0.0053%	0	0	2	-6961.814
Pro7	10	Quadratic	0.0002%	3	0	5	24.306
Pro8	2	Nonlinear	0.8587%	0	0	2	-0.095825
Pro9	7	Nonlinear	0.5182%	0	0	4	680.630
Pro10	8	Linear	0.0005%	3	0	3	7049.248
Pro11	2	Quadratic	0.0000%	0	1	0	0.750
Pro12	3	Quadratic	0.0197%	0	0	9 ³	-1.0000
Pro13	5	Nonlinear	0.0000%	0	3	0	0.0539498

of objective functions (e.g., linear, nonlinear and quadratic), but also a wide range of constraint functions (e.g., linear inequality (LI), nonlinear equality (NE), and nonlinear inequality (NI)). The feasible ratio ρ is determined by calculating the percentage of feasible solutions among 1,000,000 randomly generated points in the box constraint, i.e.,

$$\rho = |\Omega|/|X|,$$

where $|X| (= 1,000,000)$ is the number of points randomly generated from X , $|\Omega|$ is the number of feasible points out of these $|X|$ solutions. It can be observed that the feasible ratio of Problem 3, 5, 11, 13 are all zero. This is because only equality constraints applied in these problems. However, regarding Problem 2, almost (99.9962%) among the generated points are feasible points. Note that Problem 2, 3, 8 and 12 are maximization problems, and the others are minimization problems. We transform the maximization problems into minimization problems using $-f(x)$ in this study.

In order to measure the success rates of all test algorithms, we introduce the following criterions,

$$\frac{f^* - f^{**}}{|f^{**}| + 1} < \epsilon, \quad (14)$$

and

$$\frac{F^* - f^{**}}{|f^{**}| + 1} < \epsilon, \quad (15)$$

where f^* , F^* and f^{**} are the optimal value of objective function, optimal value of auxiliary function and the current known best optimal solution, respectively. ϵ is a threshold number which, in our test problems, is $10^{-2} \sim 10^{-3}$. We use the criterions (14) and (15) together because they not only guarantee optimal solution but also convergence of penalty function. For each test case, 100 independent trails of GAHJ on both Model 1 (Problem (9)) and Model 2 (Problem (10)) are executed. In order to compare the numerical performance of GAHJ on Model 1 and Model 2, we use the same parameters for both models. More specifically, the maximal generation time is set as $400n$; population size is set as $20n$; crossover rate and mutation rate are set as 0.7 and 0.1, respectively.

TABLE 5. Success rate of GAHJ

Pro.	Dim.	Model 1		Model 2	
		μ	s	μ	s
Pro1	13	10^5	100%	10^5	100%
Pro2	20	10	16%	10^4	0%
Pro3	10	10^4	100%	10^4	100%
Pro4	5	10^3	100%	10^5	36%
Pro5	4	100	100%	100	32%
Pro6	2	10^4	100%	10^4	100%
Pro7	10	10^5	100%	100	24%
Pro8	2	10^6	90%	10^6	79%
Pro9	7	10^3	100%	10^6	100%
Pro10	8	10^5	100%	10^6	12%
Pro11	2	10^4	100%	10^6	39%
Pro12	3	10^6	100%	10^5	100%
Pro13	5	10	100%	10^3	100%

Table 5 shows the success rates of GAHJ of solving these test problems by applying Model 1 and Model 2, respectively. In this table, μ and s are penalty parameter and success rate, respectively. This table clearly shows that quadratic penalty transformation of constraints (Model 1) enjoys a better success rate than that of obtained by exact penalty transformation of constraints (Model 2). However, Problem 2 is an exception. The exterior penalty function model suffered a low success rate (16%). Furthermore, the exact penalty function model even failed to obtain an optimal solution. It happens in some other literatures, like [29]. The success rate of Problem 8 for Model 2 is 90% which is less than the others. Regarding the choice of penalty parameters, the penalty parameter in Model 1 is smaller than that for Model 2.

Table 6 shows statistical results of Problem 1-13 solving by GAHJ through Model 1 and Model 2, respectively. In the table, we list the best function values (f^*), mean function values ($\bar{f}$), and the worst function values ($\tilde{f}$) out of 100 independent executions. The corresponding penalty function values (p^* , $\bar{p}$, $\tilde{p}$) are also presented. The last column is the standard deviation (S.D.) of function value out of 100 independent executions. This table shows that, except Problem 2, all of them obtained good results. Among them, Problem 1, 6, 10 and 13 are better solved by exact penalty transformation of constraints (Model 2); Problem 3, 4, 5, 7, 9 and 11 are better solved by quadratic penalty transformation (Model 1); and the same solutions are achieved for Problem 8 and 12. It is important to note that this comparison is based on the ignoring of penalty term. Technically, some of the solutions are corresponding to infeasible points, but very close to feasible region (the degree of approach can be seen from p^* , $\bar{p}$ and $\tilde{p}$). Since we use exterior penalty idea to handle the constraints, this difficulty cannot be removed in general. With respect to the standard deviation, Model 2 is better than Model 1 for Problem 1, 4, 6, 8, 10, 11 and 13, which means that Model 2 is more robust and stable than Model 1 for these problems. However, Model 1 is more stable than Model 2 for Problem 2, 3, 5, 7 and 9. For Problem 12, Model 1 and Model 2 share the same stability.

Figure 4.2 tracks the variation of objective function values of Problem 1 and Problem 12 when solved in both models. For Problem 1, objective function values

TABLE 6. Numerical results of Problem 1-13 solved by GAHJ

Pro.	Model	Best		Mean		Worst		S.D.
		f^*	p^*	$\bar{f}$	$\bar{p}$	$\underline{f}$	$\underline{p}$	
Pro1	Model 1	-15.000	1.4975e-11	-14.968331	1.5500e-11	-13.828132	3.3923e-11	2.8e-14
	Model 2	-15.000	0	-15.000	0	-15.000	0	0
Pro2	Model 1	-0.6114226	4.8896e-15	-0.5153114	5.7896e-16	-0.3924318	0	3.2e-2
	Model 2	-0.6238671	3.8964e-15	-0.4936823	2.7368e-16	-0.3492111	-1.0225e-16	5.4e-2
Pro3	Model 1	-1.0031596	1.5958e-9	-1.0031321	1.5673e-9	-1.0030899	1.5250e-9	9.5e-6
	Model 2	-0.999823	9.6033e-14	-0.9939532	1.0173e-11	-0.9814305	5.6362e-12	5.4e-3
Pro4	Model 1	-30665.547	8.8329e-11	-30605.395	1.3680e-11	-30369.105	0	19
	Model 2	-30665.541	3.2513e-8	-30665.539	2.3496e-9	-30665.537	1.3853e-9	5.3e-4
Pro5	Model 1	5126.1615	0.1704e-4	5127.8993	0.1674e-4	5160.0261	0.1915e-4	4.4
	Model 2	5126.5028	7.5138e-4	5139.9054	3.1882e-3	5173.3279	1.1084e-3	13
Pro6	Model 1	-6961.8121	0	-6961.6407	4.7569e-4	-6960.9237	0	1.5
	Model 2	-6961.8134	0	-6961.6189	7.1054e-16	-6961.0593	0	0.16
Pro7	Model 1	24.303714	1.4521e-6	24.308487	1.3796e-6	24.349013	1.2417e-6	9.4e-3
	Model 2	24.318635	0	24.422653	0	24.556940	0	6.7e-2
Pro8	Model 1	-0.0958250	0	-0.0958250	0	-0.0958250	0	2.1e-10
	Model 2	-0.0958250	0	-0.0958250	0	-0.0958250	0	1.5e-10
Pro9	Model 1	680.55806	3.6152e-3	680.55836	3.5841e-3	680.55860	3.5606e-3	1.1e-4
	Model 2	680.66750	0	681.37301	0	686.35024	0	0.7
Pro10	Model 1	7053.0718	6.5444e-10	7090.2136	1.1279e-9	7118.7250	0	19
	Model 2	7052.8082	0	7085.3769	6.8148e-6	7113.0948	0	18
Pro11	Model 1	0.7500150	1.8339e-9	0.7551281	5.4956e-9	0.7673083	2.3549e-9	4.8e-3
	Model 2	0.7503593	1.7071e-11	0.7533152	2.3821e-9	0.7604427	3.5e-3	5.3e-4
Pro12	Model 1	-1	0	-1	0	-1	0	0
	Model 2	-1	0	-1	0	-1	0	0
Pro13	Model 1	0.0540487	1.1889e-5	0.0562050	3.6475e-5	0.0639856	1.4242e-4	2.7e-3
	Model 2	0.0539679	3.8797e-5	0.0561241	3.9393e-5	0.0629757	1.7103e-5	2.2e-3

(a) Problem 1

(b) Problem 12

FIGURE 3. The Objective function value versus iteration time.

are actually smaller than the real optimal function values at the early stage. This is because the search is starting from the outside of feasible region. In this case, the value of penalty function multiplied by penalty parameter is very large, which enlarges the value of auxiliary function. Whereas, for Problem 12, most of the search appears at the inside of feasible region, which means that penalty function term is zero. So, value of auxiliary function equals value of objective function. In both

problems, the results obtained by solving Model 1 are better than those obtained by Model 2.

4.3. Comparison with FSA method. FSA method in [16] is a hybrid method to solve the constrained global optimization problem in which the simulated-annealing method is used for the global search and a direct search method is used for the local search. Thus, it is highly similar to our method. In [16], some benchmarks are tested by FSA and the results show that FSA is much better than some global optimization methods, such as Homomorphous Mappings method [19], Stochastic Ranking method [27], Adaptive Segregational Constraint Handling EA method [3] and Simple Multimembered Evolution Strategy method [24]. In view of its superiority and the similar feature to our method, we will compare our proposed method with FSA in this subsection. To apply FSA method, a constrained optimization problem in [16] is first reformulated as a form of optimizing two functions, the objective function and the constraint violation function. Then, the FSA method is applied to solve the reformulated problem. Another feature for both GAHJ and FSA method is that they are both derivative-free methods. More specifically, their requirements for both objective function and constrained functions are only Lipschitz continuous, not differentiable. The main difference between GAHJ and FSA is that GAHJ is based on genetic algorithm which is a population-based method, while FSA is based on simulated annealing method which is a point-to-point method. Another difference is the reformulation of the studied constrained optimization problem. Instead of using penalty function method, the constraints in [16] are reformulated as an nonnegative objective, named as constrained violation function. If a point is feasible, the value of constrained violation function equals to 0. Otherwise, it is positive. Under this strategy, the original constrained global optimization problem has been transformed into a multi-objective optimization problem.

In the following tables (Table 7 and Table 8), the data of FSA is taken from Table 2 of reference [16] and the data of GAHJ is taken the better one of Model 1 and Model 2.

Table 7 illustrated the best optimal solutions and the worst optimal solutions obtained by FSA and GAHJ. From the data showed in the table, for Problem 8 and 12, FSA and GAHJ obtained the same results, which are the best known optimal solutions. For Problems 1, 4, 5, 6, 7, 9 and 10, all of the best and the worst optimal solutions obtained by GAHJ are better than those obtained by FSA. For Problem 3, the best optimal solution obtained by GAHJ is -1.0031596 which is not as good as FSA (-1.0000015), but the worst optimal solution obtained by GAHJ is -1.0030899 which is much more better than FSA (-0.9915186). For Problem 2, FSA performs better at the best optimal solution, but weaker at the worst optimal solution. For Problem 13, there is a big gap between the best optimal solution and the worst optimal solution obtained by FSA. The worst optimal solution obtained by GAHJ is much more better than that obtained by FSA.

In order to investigate the robustness of GAHJ, we compare its statistic performances with those of FSA. Table 8 reports mean value (Mean) and standard division (S.D.) of optimal solutions for both GAHJ and FSA. Table 8 shows that GAHJ achieves better mean values for Problem 1, 2, 3, 4, 7, 9, 10, 11 and 13. For Problem 10, the mean value of FSA is 7509.32104, while that of GAHJ is 7085.3769 which is much smaller than that of FSA. For Problem 13, the mean value of FSA is 0.2415963, while that of GAHJ is 0.0539498. FSA achieves a better mean value than that of GAHJ only for Problems 5, 6 and 11. On the other hand, from data of

TABLE 7. Comparison between GAHJ and FSA

Pro.	Optimal solution	FSA	GAHJ
Pro1	Best	-14.999	-15.000
	Worst	-14.980	-15.000
Pro2	Best	-0.7549125	-0.62386713
	Worst	-0.2713110	-0.3492111
Pro3	Best	-1.0000015	-1.0031596
	Worst	-0.9915186	-1.0030899
Pro4	Best	-30665.5380	-30665.541
	Worst	-30664.6880	-30665.537
Pro5	Best	5126.4981	5126.1615
	Worst	5126.4981	5160.0261
Pro6	Best	-6961.81388	-6961.8134
	Worst	-6961.81388	-6961.0593
Pro7	Best	24.310571	24.303714
	Worst	24.644397	24.349013
Pro8	Best	-0.095825	-0.095825041
	Worst	-0.095825	-0.095825040
Pro9	Best	680.63008	680.55806
	Worst	680.69832	680.55860
Pro10	Best	7059.86350	7052.8082
	Worst	9398.64920	7113.0948
Pro11	Best	0.7499990	0.75001507
	Worst	0.7499990	0.76730836
Pro12	Best	-1.0000	-1.0000
	Worst	-1.0000	-1.0000
Pro13	Best	0.0539498	0.053967965
	Worst	0.4388511	0.062975713

TABLE 8. Statistic comparison between GAHJ and FSA

Pro.	Statistic features	FSA	GAHJ
Pro1	Mean	-14.993316	-15.000
	S.D.	0.004813	0.000000
Pro2	Mean	0.3717081	-0.5153114
	S.D.	0.098023	0.032000
Pro3	Mean	-0.9991874	-1.0031321
	S.D.	0.001653	0.000009
Pro4	Mean	-30665.4665	-30665.539
	S.D.	0.173218	0.0000530
Pro5	Mean	5126.4981	5127.8993
	S.D.	0.000000	4.4
Pro6	Mean	-6961.81388	-6961.6407
	S.D.	0.000000	1.5
Pro7	Mean	24.3795271	24.308487
	S.D.	0.071635	0.009400
Pro8	Mean	-0.095825	-0.0958250
	S.D.	0.000000	0.000000
Pro9	Mean	680.63642	680.55836
	S.D.	0.014517	0.000110
Pro10	Mean	7509.32104	7085.3769
	S.D.	542.3421	18
Pro11	Mean	0.7499990	0.7533152
	S.D.	0.000000	0.00053
Pro12	Mean	-1.0000	-1.0000
	S.D.	0.000000	0.000000
Pro13	Mean	0.2977204	0.0561241
	S.D.	0.188652	0.002200

standard division, the optimal solutions obtained by GAHJ are more robust than those obtained by FSA. Except Problems 5, 6 and 11, solutions obtained by GAHJ have smaller standard divisions than those obtained by FSA, which means that

GAHJ is more robust than FSA. GAHJ and FSA share the same statistic performances for Problems 8 and 12. The above analysis clearly shows that as a global optimization solver, GAHJ is much better than FSA.

5. Conclusion. In this paper, we developed a new hybrid method to solve a class of constrained global optimization problems. This method is based on the combination of the advantages of global exploration of genetic algorithm and the local exploitation of Hooke-Jeeves method. More precisely, The Hooke-Jeeves method was embedded into genetic algorithm as an acceleration operator during the iterations. The numerical experiments show that our proposed method achieves better performances than genetic algorithm, Hooke-Jeeves method and some available global optimization solvers. However, since we use exterior penalty function method to handle the constraints, some of the optimal solutions obtained by the proposed method may be infeasible. Therefore, our future work for this subject is to improve the constraint handling technique to ensure the feasibility of the obtained solution. A possible strategy is to introduce the greedy selection (death penalty) [4]. This strategy excludes all the infeasible candidate solutions gradually.

REFERENCES

- [1] Adil M Bagirov and Asef Nazari Ganjehlou. A quasisecant method for minimizing nonsmooth functions. *Optimization Methods & Software*, 25(1):3–18, 2010.
- [2] Sherali H.D. Shetty C. M. Bazaraa, M. S. *Nonlinear programming: theory and algorithm (Third Edition)*. Wiley Online Library, 2006.
- [3] S Ben Hamida and Marc Schoenauer. Aschea: New results using adaptive segregational constraint handling. In *Evolutionary Computation, 2002. CEC'02. Proceedings of the 2002 Congress on*, volume 1, pages 884–889. IEEE, 2002.
- [4] Zixing Cai and Yong Wang. A multiobjective optimization-based evolutionary algorithm for constrained optimization. *Evolutionary Computation, IEEE Transactions on*, 10(6):658–675, 2006.
- [5] G.D. Camp. Inequality-constrained stationary-value problems. *Journal of the Operations Research Society of America*, 3(4):548–550, 1955.
- [6] C.W. Carroll and A.V. Fiacco. The created response surface technique for optimizing nonlinear restrained systems. *Operations Research*, pages 169–185, 1961.
- [7] R. Chelouah and P. Siarry. A hybrid method combining continuous tabu search and nelder-mead simplex algorithms for the global optimization of multim minima functions. *European Journal of Operational Research*, 161(3):636–654, 2005.
- [8] Zhongwen Chen, Songqiang Qiu, and Yujie Jiao. A penalty-free method for equality constrained optimization. *Journal of Industrial and Management Optimization*, 9(2):391–409, 2013.
- [9] Frank E Curtis and Michael L Overton. A sequential quadratic programming algorithm for nonconvex, nonsmooth constrained optimization. *SIAM Journal on Optimization*, 22(2):474–500, 2012.
- [10] N. Durand and J.M. Alliot. A combined nelder-mead simplex and genetic algorithm. In *Proceedings of the Genetic and Evolutionary Computation Conference GECCO*, volume 99, pages 1–7, 1999.
- [11] R. Fletcher. An ideal penalty function for constrained optimization. *IMA Journal of Applied Mathematics*, 15(3):319–342, 1975.
- [12] David E Goldberg. Genetic algorithms in search, optimization, and machine learning. 1989.
- [13] H.J. Greenberg. The generalized penalty-function/surrogate model. *Operations Research*, pages 162–178, 1973.
- [14] A.R. Hedar. Global optimization methods and codes. <http://www-optima.amp.i.kyoto-u.ac.jp/member/student/hedar/Hedar.html>.
- [15] A.R. Hedar and M. Fukushima. Hybrid simulated annealing and direct search method for nonlinear global optimization. *Department of Applied Mathematics & Physics Kyoto University*, pages 2001–013, 2001.
- [16] A.R. Hedar and M. Fukushima. Derivative-free filter simulated annealing method for constrained continuous global optimization. *Journal of Global Optimization*, 35(4):521–549, 2006.
- [17] Elizabeth Karas, Ademir Ribeiro, Claudia Sagastizábal, and Mikhail Solodov. A bundle-filter method for nonsmooth convex constrained optimization. *Mathematical Programming*, 116(1-2):297–320, 2009.
- [18] Napsu Karmitsa, Adil Bagirov, and MM Mäkelä. Comparing different nonsmooth minimization methods and software. *Optimization Methods and Software*, 27(1):131–153, 2012.
- [19] Slawomir Koziel and Zbigniew Michalewicz. Evolutionary algorithms, homomorphous mappings, and constrained parameter optimization. *Evolutionary computation*, 7(1):19–44, 1999.
- [20] Oliver Kramer. A review of constraint-handling techniques for evolution strategies. *Applied Computational Intelligence and Soft Computing*, 2010, 2010.
- [21] Yanqun Liu. An exterior point linear programming method based on inclusive normal cone. *Journal of Industrial and Management Optimization*, 6(4):825–846, 2010.
- [22] D.G. Luenberger. Introduction to linear and nonlinear programming. 1973.
- [23] Rammohan Mallipeddi and Ponnuthurai N Suganthan. Ensemble of constraint handling techniques. *Evolutionary Computation, IEEE Transactions on*, 14(4):561–579, 2010.
- [24] Efrén Mezura-Montes and CA Coello Coello. A simple multimembered evolution strategy to solve constrained optimization problems. *Evolutionary Computation, IEEE Transactions on*, 9(1):1–17, 2005.

- [25] Wataru Nakamura, Yasushi Narushima, and Hiroshi Yabe. Nonlinear conjugate gradient methods with sufficient descent properties for unconstrained optimization. *Journal of Industrial and Management Optimization*, 9(3):595–619, 2013.
- [26] W.P. Pierskalla. Mathematical programming with increasing constraint functions. *Management Science*, pages 416–425, 1969.
- [27] Thomas P. Runarsson and Xin Yao. Stochastic ranking for constrained evolutionary optimization. *Evolutionary Computation, IEEE Transactions on*, 4(3):284–294, 2000.
- [28] JA Vasconcelos, JA Ramirez, RHC Takahashi, and RR Saldanha. Improvements in genetic algorithms. *Genetics, IEEE Transactions on*, 37(5):3414–3417, 2001.
- [29] Y. Wang, Z. Cai, Y. Zhou, and Z. Fan. Constrained optimization based on hybrid evolutionary algorithm and adaptive constraint-handling technique. *Structural and Multidisciplinary Optimization*, 37(4):395–413, 2009.
- [30] Changjun Yu, Kok Lay Teo, Liansheng Zhang, and Yanqin Bai. A new exact penalty function method for continuous inequality constrained optimization problems. *Journal of Industrial and Management Optimization*, 6(4):895, 2010.
- [31] Qiying Hu Zhiqiang Meng and Chuangyin Dang. A penalty function algorithm with objective parameters for nonlinear mathematical programming. *Journal of Industrial and Management Optimization*, 5(3):585–601, 2009.

E-mail address: q.long@student.ballarat.edu.au

E-mail address: changzhiwu@yahoo.com