Applied Soft Computing 47 (2016) 281–294

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

A hybrid approach to constrained global optimization☆

Jianjun Liu[a], Shaohua Zhang[b,*], Changzhi Wu[c], Jingwei Liang[a], Xiangyu Wang[c,d], Kok Lay Teo[e]

[a] *College of Science, China University of Petroleum, Beijing 102249, China*
[b] *Shanghai Development Center of Computer Software Technology, Shanghai, China*
[c] *Australasian Joint Research Centre for Building Information Modelling, School of Built Environment, Curtin University, Perth, WA 6845, Australia*
[d] *Department of Housing and Interior Design, Kyung Hee University, Seoul, Republic of Korea*
[e] *Department of Mathematics, Curtin University, Perth, WA 6845, Australia*

ARTICLE INFO

Article history:
Received 9 March 2015
Received in revised form 10 May 2016
Accepted 11 May 2016
Available online 9 June 2016

Keywords:
Global optimization
Exact penalty
Greedy Diffusion Search
LBFGS method

ABSTRACT

In this paper, we propose a novel hybrid global optimization method to solve constrained optimization problems. An exact penalty function is first applied to approximate the original constrained optimization problem by a sequence of optimization problems with bound constraints. To solve each of these box constrained optimization problems, two hybrid methods are introduced, where two different strategies are used to combine limited memory BFGS (L-BFGS) with Greedy Diffusion Search (GDS). The convergence issue of the two hybrid methods is addressed. To evaluate the effectiveness of the proposed algorithm, 18 box constrained and 4 general constrained problems from the literature are tested. Numerical results obtained show that our proposed hybrid algorithm is more effective in obtaining more accurate solutions than those compared to.

1. Introduction

Many practical problems can be formulated as optimization problems [1,2]. A general optimization problem can be stated as follows:

$$\min \quad f(x) \tag{1}$$

$$\text{s.t.} \quad h_i(x) = 0, \quad i = 1, \ldots, l, \tag{2}$$

$$g_j(x) \le 0, \quad j = 1, \ldots, m, \tag{3}$$

$$x \in X = \{x \in \mathbb{R}^n : L \le x \le U\}, \tag{4}$$

where $x \in \mathbb{R}^n$, f, h_i, $i = 1, \ldots, l$, and g_j, $j = 1, \ldots, m$, are continuously differentiable functions, $L = [L_1, L_2, \ldots, L_n]$ and $U = [U_1, U_2, \ldots, U_n]$ are, respectively, the lower and upper bounds. Let this problem be referred to as Problem (P). To proceed further, we suppose that this problem has at least one feasible solution.

In many real world applications, Problem (P) is non-convex, either due to the non-convexity of the objective function or the constraint functions. For such a case, Problem (P) may admit many local minima. In practice, local minima are useless if their corresponding objective function values are far away from the global minimum [3]. Thus, it is important to develop methods for finding a global minimum of Problem (P). In the past several decades, there have been extensive efforts dedicated to global optimization. In general, global optimization methods can be classified into three main categories: deterministic methods, stochastic methods and hybrid methods. For the methods belonging to the first category, they are developed based on deterministic search strategies in which only deterministic information is involved for both local and global searches. In particular, for each of these methods, it relies heavily on the construction of an auxiliary function to escape from local minima, such as tunnelling function [4] and filled function [5,6], where there are several parameters to be adjusted. Tuning these parameters is computationally expensive. For the methods belonging to the second category, probabilistic techniques are utilized to escape from local minima, such as Genetic Algorithm [7–9], Ant Colony Optimization [10,11], Simulated Annealing algorithm [12], Artificial Bee Colony algorithm [13–15], Particle Swarm Optimization [16,17], Collective neuro-dynamic optimization [18], Artificial algae algorithm [19] and Differential search algorithm [20,21]. However, these methods tend to obtain solution with low accuracy and are computationally expensive due to lack of guidance by gradient during the searching process [22]. Their performances are poor in terms of convergence [23].

☆ The authors gratefully acknowledge the financial support from Australian Research Council (LP140100873), the National Natural Science Foundation of China (Grant Nos. 11371371 and 61473326) and the Science Foundation of China University of Petroleum, Beijing (2462015QZDX02).

* Corresponding author.
E-mail address: zsh@ssc.stn.sh.cn (S. Zhang).

http://dx.doi.org/10.1016/j.asoc.2016.05.021
1568-4946/© 2016 Elsevier B.V. All rights reserved.

The methods belonging to the third category are known as hybrid methods, where some stochastic schemes are combined together or population based search methods are combined with deterministic methods so as to speed up convergence process. In [24], Harmony Search (HS) and Artificial Bee Colony (ABC) algorithm are combined together to solve a class of box-constrained optimization problems in which ABC is incorporated to improve the local convergence of HS. In [23], a hybrid optimization technique is proposed through combining a genetic algorithm with a local search strategy based on the interior point method. In [25], an improved genetic algorithm (IGA) and an improved particle swarm optimization (IPSO) algorithm are combined and applied to optimize the amplitude of the current excitation of the spherical conformal array. In [26,3], Simulated Annealing method is used to escape from local optima obtained by gradient-based deterministic method. In [27], ABC algorithm is combined with a modified pattern search method to improve success rate and solution accuracy for box constrained optimization problems. In [28], Particle Swarm Optimizer is combined with BFGS to solve box-constrained optimization problems, where BFGS is for the local search. In [22], this hybrid method is further developed to solve general constrained optimization problems. In [29], evolutionary computation (EC) algorithms are combined with a sequential quadratic programming (SQP) algorithm to solve constrained global optimization problems. The hybrid methods mentioned above have better numerical performances when compared with pure stochastic search methods. In these hybrid methods, the stochastic methods are mainly utilized to help obtain a better initial condition for further local minimizing which means that only exploration search is used. Note that the original stochastic methods are designed not only for exploration search, but also for exploitation search. For a hybrid algorithm, if a gradient-based method is embedded for local search, the exploration would be strengthened at the expense of weakening exploitation. However, the performance of these algorithms depends heavily on tuning parameters in the stochastic algorithms. If the parameters are not tuned appropriately, the solution obtained will still be trapped into local minimum. To overcome this drawback, Dynamically Dimensioned Search Algorithm is developed in [30] where no parameters tuning is required. However, that method is a single-solution based heuristic global search algorithm. Stochastic based search methods are applicable only to unconstrained or box-constrained optimization methods, and hence are not directly applicable to solve Problem (P) which is a constrained optimization problem involving both equality and inequality constraints. In the literature, a constrained optimization problem is often transformed into a box constrained optimization problem by augmenting the constraint functions to the cost function using the augmented Lagrangian penalty method [22]. However, the penalty parameter is required to go to infinity for achieving feasibility. In this paper, the exact penalty function method (EPM) (see [31,32]) will be applied to convert the constrained optimization problem (P) into a box constrained optimization problem. A major advantage of this approach is that the penalty parameter needs only to be greater than or equal to some finite value for achieving feasibility. Then, a new population-based stochastic search method, called the Greedy Diffusion Search (GDS), is proposed to solve the box constrained optimization problems where two parameters are included. In our extensive experimental experiences, both of the two parameters can be pre-set without affecting performance and thus, no parameters tuning is required in GDS. In addition, the convergence issue is addressed. However, this method is strong in exploration but suffers from poor exploitation. Thus, the limited memory BFGS is embedded into GDS in two different strategies to improve its exploitation. An effective new hybrid search method is thus obtained for solving Problem (P).

The rest of this paper is organized as follows. In Section 2, an exact penalty method is introduced to tackle the constraints. In Section 3, two hybrid methods are proposed. Numerical results and comparisons between different methods are reported in Section 4. Section 5 concludes the paper.

2. Exact penalty function method (EPM)

Nonlinear constrained optimization problems can be solved through solving a sequence of box-constrained optimization problems by augmenting the constraint functions to the objective function using the penalty function method [6,33], to form an augmented objective function. For optimization problems with equality and inequality constraints, the penalty parameter in the augmented objective function is, in principle, required to go to infinity for achieving feasibility of the solution obtained. However, this is clearly undoable. On the other hand, the exact penalty function method introduced in [31,32] does not require the penalty parameter to go to infinity [31,32] for achieving feasibility of the solution obtained. In what follows, the exact penalty function approach proposed in [31] will be briefly described.

Let us first define the constraint violation function on X as follows:

$$G(x) = \sum_{i=1}^{l}[h_i(x)]^2 + \sum_{j=1}^{m}[\max\{g_j(x), 0\}]^2. \tag{5}$$

It is clear that $G(x) = 0$ if and only if x satisfies the equality constraints (2) and the inequality constraints (3). Furthermore, $G(x)$ is a continuously differentiable function [31].

For a given $\bar{\epsilon} > 0$, we define the following penalty function on $X \times [0, \bar{\epsilon}]$:

$$F_\sigma(x,\epsilon) = \begin{cases} f(x), & \text{if} \quad \epsilon = 0,\, G(x) = 0; \\ f(x) + \epsilon^{-\alpha}G(x) + \sigma\epsilon^{\beta}, & \text{if} \quad \epsilon \in (0, \bar{\epsilon}]; \\ \infty, & \text{if} \quad \epsilon = 0,\, G(x) \neq 0; \end{cases} \tag{6}$$

where $\sigma > 0$ is a penalty parameter, α and β are two positive constants satisfying $1 \le \beta \le \alpha$.

Instead of solving Problem (P) directly, let us consider the following optimization problem:

$$\min_{(x,\epsilon)\in X\times[0,\bar{\epsilon}]} F_\sigma(x,\epsilon). \tag{7}$$

Let this problem be referred to as Problem (P_σ). For a given σ, minimizing $F_\sigma(x, \epsilon)$ with respect to $(x, \epsilon) \in X \times [0, \bar{\epsilon}]$ is equivalent to minimizing $f(x) + \epsilon^{-\alpha}G(x) + \sigma\epsilon^{\beta}$. Thus, if σ is increased, ϵ^{β} will be decreased. Hence, the constraint violation $G(x)$ will be decreased. Therefore, the increase of the penalty parameter σ will eventually yield a feasible solution.

The two theorems in Appendix A reveal the relationship between Problem (P) and Problem (P_σ).

Theorem 2 in Appendix A shows that there exists a threshold $\bar{\sigma}$, such that for all $\sigma \ge \bar{\sigma}$, any local solution of Problem (P_σ) is also a local solution of Problem (P). This important property is not shared by the augmented Lagrangian penalty method [28], for which the penalty parameter is, in principle, required to go to infinity ensuring feasibility of the solution obtained. Since global solutions are included in local solutions, a global solution of Problem (P_{σ_k}) will yield a global solution of Problem (P). From this observation together with Theorems 1 and 2 in Appendix A, the exact penalty method (EPM) is utilized to convert Problem (P) into Problem (P_σ). In Section 3, an algorithm is proposed to solve Problem (P) through solving a sequence of Problem (P_σ). This algorithm is referred to as Algorithm 1.

To continue, we denote, for notational simplicity, $z = (x, \epsilon)$ and $\Omega = X \times [0, \bar{\epsilon}]$. Then, Problem (P_σ) can be written as:

$$\min_{z \in \Omega} F_\sigma(z). \tag{8}$$

3. Two hybrid methods for Problem (P_σ)

Algorithm 1. Exact penalty method (EPM) for Problem (P)

```
Initialization:
    Initialize $\sigma_0 > 0$, maximum penalty parameter $\sigma_{\max}$, scale factor $\varsigma > 1$ and tolerance $\varepsilon_1 > 0$; Set $k = 0$.
Iteration:
    1: Use Algorithm L-GDS or L-RGDS to solve Problem $(P_{\sigma_k})$ and output the optimal solution $(x^{k,*}, \epsilon^{k,*})$.
    2: If $\epsilon^{k,*} \le \varepsilon_1$, then stop and output $x^{k,*}$ as the optimal solution of Problem $(P)$; otherwise, set $\sigma_{k+1} = \varsigma\sigma_k$.
    3: If $\sigma_{k+1} \le \sigma_{\max}$, set $k = k + 1$ and go to Step 1. Otherwise, stop and output "algorithm cannot find a solution of Problem $(P)$".
```

Note: Algorithm L-GDS and Algorithm L-RGDS in Algorithm 1 will be defined in Section 3.3.

Two hybrid methods will be introduced to solve Problem (P_σ). These two hybrid methods are constructed based on two different strategies of combining limited memory BFGS (L-BFGS) and a novel stochastic search method. Let us briefly introduce L-BFGS as reported in [34].

3.1. Limited memory BFGS method

L-BFGS is an adaptation of the BFGS method for large-scale problems. For the box-constrained optimization problem, we define the following projection.

$$P_\Omega(z_i) = \begin{cases} \underline{z}_i, & \text{if} \quad z_i < \underline{z}_i; \\ z_i, & \text{if} \quad \underline{z}_i \le z_i \le \bar{z}_i; \\ \bar{z}_i, & \text{if} \quad z_i > \bar{z}_i, \end{cases} \tag{9}$$

where z_i is the i-th element of z, $\underline{z}_i = L_i, i = 1, \ldots, n$, $\underline{z}_{n+1} = 0$, $\bar{z}_i = U_i, i = 1, \ldots, n$, and $\bar{z}_{n+1} = \bar{\epsilon}$.

L-BFGS has superlinear convergence, and requires less storage than BFGS. Thus, L-BFGS has attracted considerable attention and has been used to solve many practical problems. For further information, see [35–37] for general methods and [38,39] for applications.

As the original BFGS, L-BFGS uses an approximation to the Hessian matrix to steer its search through variable space. BFGS stores a dense approximation to the inverse Hessian while L-BFGS stores only a few vectors that represent the approximation implicitly. Instead of updating the inverse Hessian H_k at every iteration, L-BFGS maintains a history of the past $\widehat{m}$ updates and then updates, where the history size $\widehat{m}$ is generally small. L-BFGS starts with an initial point $z^{(0)} \in \Omega$ and generates iterations $z^{(k+1)}$ by the process $z^{(k+1)} = z^{(k)} + \lambda_k p^{(k)}$, where $p^{(k)}$ is the direction vector and $\lambda_k \ge 0$ is a step-length, usually chosen in such a way that it satisfies the Wolfe inexact line search conditions (see [1])

$$F_\sigma(z^{(k+1)}) - F_\sigma(z^{(k)}) \le \zeta\lambda_k g_k^T p^{(k)}, g_{k+1}^T p^{(k)} \ge \eta g_k^T p^{(k)} \tag{10}$$

where $0 < \zeta < \frac{1}{2}, \zeta < \eta < 1$, $g_k = \nabla F_\sigma(z^{(k)})$, and $p^{(k)} = - H_k g_k$ with a symmetric positive definite matrix H_k. Usually H_0 is a multiple of I and H_{k+1} is obtained from H_k by a variable metric update to satisfy the quasi-Newton condition

$$H_{k+1} y_k = s_k \tag{11}$$

where $s_k = z^{(k+1)} - z^{(k)} = \lambda_k p^{(k)}$ and $y_k = g_{k+1} - g_k$. Then, we have

$$H_{k+1} = V_k^T H_k V_k + \rho_k s_k s_k^T \tag{12}$$

with

$$\rho_k = \frac{1}{y_k^T s_k}, V_k = I - \rho_k y_k s_k^T$$

Therefore, H_{k+1} in L-BFGS can be updated as follows:

$$\begin{aligned} H_{k+1} &= V_k^T H_k V_k + \rho_k s_k s_k^T \\ &= V_k^T [V_{k-1}^T H_{k-1} V_{k-1} + \rho_{k-1} s_{k-1} s_{k-1}^T] V_k + \rho_k s_k s_k^T \\ &= \cdots \\ &= [V_k^T \cdots V_{k-\widehat{m}+1}^T] H_{k-\widehat{m}+1} [V_{k-\widehat{m}+1} \cdots V_k] \\ &\quad + \rho_{k-\widehat{m}+1} [V_{k-1}^T \cdots V_{k-\widehat{m}+2}^T] s_{k-\widehat{m}+1} s_{k-\widehat{m}+1}^T [V_{k-\widehat{m}+2} \cdots V_{k-1}] \\ &\quad + \cdots + \rho_k s_k s_k^T. \end{aligned} \tag{13}$$

Now, we modify the L-BFGS method for unconstrained optimization in [34] to box-constrained optimization problem as given in Algorithm 2.

Algorithm 2. Limited memory BFGS (L-BFGS) for box-constrained problem

```
Initialization:
    Choose $z^{(0)} \in \Omega; H_0 \in \mathbb{R}^{(n+1)\times(n+1)}$ positive definite;
    $0 < \zeta < \frac{1}{2}; \zeta < \eta < 1; m \in \mathbb{N}^+; \varepsilon_2 \ge 0$; Set $k = 0$.
Iteration:
    1: If $||g_k|| \le \varepsilon_2$, stop.
    2: Compute $p^{(k)} = - H_k g_k$.
    3: If $\lambda_k = 1$ satisfies Eq. (10), go to the next step. Otherwise, run the Wolfe line search to find $\lambda_k$ such that Eq. (10) is satisfied.
    4: Set $z^{(k+1)} = z^{(k)} + \lambda_k p^{(k)}$.
    5: Run the projection operator given by Eq. (9).
    6: Let $\widehat{m} = \min\{k+1, m\}$. Update $H_0$ for $\widehat{m}$ times to get $H_{k+1}$ according to Eq. (13).
    7: Set $k = k + 1$.
```

It is worth mentioning that Algorithm 2 is as simple as L-BFGS for unconstrained optimization problem as described in [34] since the computation of the projection operation (9) is simple.

3.2. A new stochastic search strategy – Greedy diffusion search

Although L-BFGS has excellent performance for local search, the solution obtained is often a local optimal solution for multimodal functions. To escape from the local optimal basin, we propose a new stochastic search strategy, called *Greedy Diffusion Search* (*GDS*), as described below.

After an initial point $z^{(0)} \in \Omega$ is chosen, q_1 additional points, which constitute a generation, will be generated according to the formulas given below:

$$\widetilde{z}_i^{(l+1)} = (1 - \theta_{l,i}) z^{(l)} + \theta_{l,i} \xi_i^{(l)}, i = 1, 2, \ldots, q_1, \tag{14}$$

with

$$\theta_{l,i} = \frac{1}{1 + e^{\frac{l - tN}{a}}}, \tag{15}$$

where the search direction ξ_i is randomly generated with a uniform distribution in the box Ω, and $\theta_{l,i}$ is the step-length determined by Eq. (15). l and N are the current generation number and the maximum generation number, respectively. Since $z^{(l)} \in \Omega$ and $\xi_i^{(l)} \in \Omega$, $i = 1, \ldots, q_1$, $\widetilde{z}_i^{(l+1)} \in \Omega$. To ensure that the algorithm is having the descent property, we further choose $z^{(l+1)}$ according to the following greedy rule:

$$z^{(l+1)} = arg \min\{F_\sigma(z^{(l)}), F_\sigma(\widetilde{z}_i^{(l+1)}) : i = 1, 2, \ldots, q_1\} \tag{16}$$

i.e., $z^{(l+1)}$ is the best point among the current $\widetilde{z}_i^{(l+1)}$.

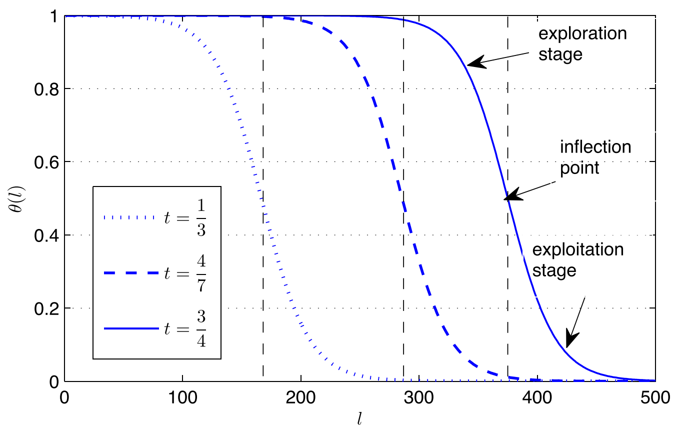

Fig. 1. θ versus l with different t.

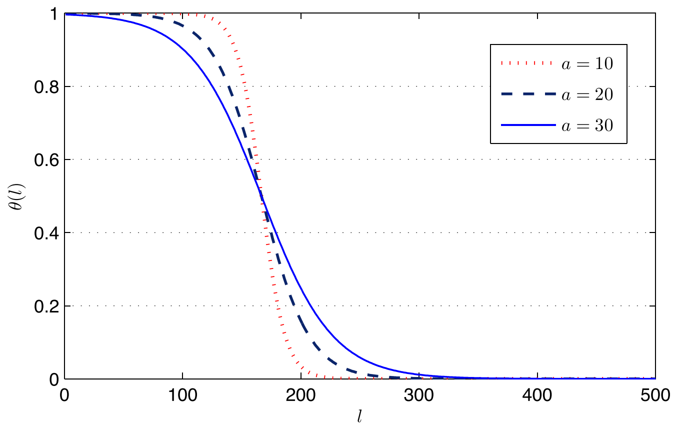

Fig. 2. θ versus l with different a.

To maintain exploration capability at a later stage, we choose q_2 points randomly with a uniform distribution in Ω, denoted by $\widehat{z}_j^{(l+1)} \in \Omega, j=1,2,\ldots,q_2$, and integrate them with the current generation in the lth iteration by Eq. (14). Generally, q_2 is chosen such that $q_2 < q_1$. Based on the greedy rule, (16) becomes:

$$z^{(l+1)} = arg\min\{F_\sigma(z^{(l)}), F_\sigma(\widetilde{z}_i^{(l+1)}), F_\sigma(\widehat{z}_j^{(l+1)}) : i=1,2,\ldots, q_1, j=1,2\ldots,q_2\} \tag{17}$$

There are two parameters in (15): the translation parameter t and the accuracy parameter a.

Translation parameter (t): This parameter reflects the relationship between step-length and iteration number. According to Eq. (15), if a is fixed, the step-length $\theta_{l,i}$ will vary with respect to t, l and the constant N. In particular, t acts as a translation scalar. Fig. 1 plots the curve of $\theta_{l,i}$ with $a=20$ for $t=\frac{1}{3}$, $\frac{4}{7}$ and $\frac{3}{4}$. This figure shows that the larger the t is, the more diverging the points are searched according to the rule given by Eq. (14).

Accuracy parameter (a): The parameter a in Eq. (15) plays an important role to determine the accuracy of the solution obtained. Fig. 2 demonstrates that if the value of a is large, then the step-length is small in earlier stage of the iteration process but becomes larger in the later stage, and vice versa. It can be preset as a constant relating to the tolerance $\varepsilon_2 \geq 0$ (for example, $a = 2\log\frac{1}{\varepsilon_2}$).

The two main features of any stochastic algorithm are exploration and exploitation. Exploration is to generate diverse solutions and exploitation is to focus on the search in a local region. At the initial stage, the step-length $\theta_{l,i}$ is large. The diffusion points $\widetilde{z}_1^{(l)},\ldots,\widetilde{z}_{q_1}^{(l)}$ from the current points as well as $\widehat{z}_1^l,\ldots,\widehat{z}_{q_2}^l$ will explore the search space. As the generation increases, the step-length $\theta_{l,i}$ will decrease. The first q_1 diffusion points $\widetilde{z}_1^{(l)},\ldots,\widetilde{z}_{q_1}^{(l)}$ will play the role of exploitation, while the last q_2 points $\widehat{z}_1^l,\ldots,\widehat{z}_{q_2}^l$ maintain the role of exploration. Since at each iteration, the best solution among q points is maintained, Algorithm GDS is a descent algorithm.

Based on the above analysis, we now formally stated GDS in Algorithm 3.

Algorithm 3. Greedy Diffusion Search (GDS)

```
Initialization:
  Choose $\widetilde{z}^{(0)} \in \Omega$; $q_1 > 0$, $q_2 > 0$, $N \in \mathbb{N}^+$; $\varepsilon_3 \geq 0$. Set $l=0$.
Iteration:
  1: while $l \leq N$ do
  2:     Generate the step-lengths $\theta_{l,i}$ satisfying Eq. (15), $i=1,\ldots,q_1$.
  3:     Calculate $\widetilde{z}_i^{(l+1)}$ according to Eq. (14), $i=1,\ldots,q_1$.
  4:     Choose $\widehat{z}_j^{(l+1)}$, $j=1,\ldots,q_2$, randomly in $\Omega$.
  5:     Output the elitist solution according to Eq. (17).
  6:     Set $l=l+1$.
  7: end while
  8: Output result.
```

The updating rule (14) in GDS is simpler than most of the existing population-based stochastic search methods, such as Genetic Algorithm, Particle Swarm Optimization, Firefly Algorithm and Artificial Bee Colony. Intuitively, the performance of GDS may be inferior to these existing methods. Through our extensive experiments, we observe that GDS is good at exploration, but weak in exploitation. To improve its exploitation, we will propose two different strategies to combine L-BFGS with GDs. Thus, two hybrid algorithms are obtained to solve Problem (P_σ).

3.3. Two hybrid algorithms

The first hybrid algorithm which is referred to as L-GDS, is to combine L-BFGS with GDS such that the exploitation capability is strengthened. This algorithm is formally stated as Algorithm 4.

Algorithm 4. Hybridizing L-BFGS with GDS (L-GDS)

```
Initialization:
  Choose $z^{(0)} \in \Omega$, $y^{(0)}=z^{(0)}$, the tolerance $\varepsilon_4>0$ and the maximum
  iteration number $K \in \mathbb{N}^+$. Set parameters in Algorithms L-BFGS and GDS.
  Set $k=1$.
Iteration:
  1: while $k \leq K$ do
  2:     Run Algorithm GDS and output $z^{(k)}$.
  3:     Take $z^{(k)}$ as an initial point and run Algorithm L-BFGS, output $y^{(k)}$.
  4:     if $||F_\sigma(y^{(k+1)}) - F_\sigma(y^{(k)})|| < \varepsilon_4$ then
  5:         Return
  6:     end if
  7:     Set $k=k+1$.
  8: end while
  9: Output result.
```

In Algorithm L-GDS, GDS is used to obtain a good local minimizer $z^{(k)}$. Then, L-BFGS is carried out to refine the local search around this local minimizer $z^{(k)}$. To jump out the current local minimizer, Algorithm GDS is carried out to obtain a better initial point for L-BFGS to be executed again. This process is repeated until the convergence is achieved.

To avoid excessive local search, the tolerance ε in L-BFGS should not be set too small. However, in some applications, an accurate solution is required. Thus, to increase accuracy while maintaining low computational burden, we propose the following algorithm, which is referred as Algorithm L-RGDS, in which GDS is triggered by a random number. If this random number is smaller than a given threshold, GDS will be skipped and L-BFGS will be in action. This algorithm is stated as Algorithm 5.

Algorithm 5. L-BFGS with random exploration search (L-RGDS)

```
Initialization:
  Choose $z^{(0)} \in \Omega$, the tolerance $\varepsilon_4 > 0$, the maximum iteration number
  $K \in \mathbb{N}^+$ and a random constant $r_0 \in [0, 1]$; Set parameters in Algorithms
  L-BFGS and GDS. set $k = 1$.
Iteration:
  1: while $k \le K$ do
  2:   Generate a random number $0 < r < 1$.
  3:   if $r > r_0$ then
  4:     Run Algorithm GDS and output $z^{(k)}$.
  5:   else
  6:     Set $z^{(k)} = y^{(k)}$
  7:   end if
  8:   Take $z^{(k)}$ as an initial point and run Algorithm L-BFGS, output $y^{(k)}$.
  9:   if $||F_\sigma(y^{(k+1)}) - F_\sigma(y^{(k)})|| \le \varepsilon_4$ then
  10:     Return
  11:   end if
  12:   Set $k = k + 1$.
  13: end while
  14: Output result.
```

3.4. Convergence of the two algorithms

Based on some appropriate assumptions, the sequence $z^{(k)}$ obtained by L-GDS (respectively, L-RGDS) converges with probability one to the region Ω_ε for any $\varepsilon > 0$, where $\Omega_\varepsilon = \{z \in \Omega : |F_\sigma(z) - F_\sigma(z^*)| \le \varepsilon\}$. Moreover, Algorithm L-GDS (respectively, L-RGDS) converges with probability one to a global minimum. Several convergence theorems and proofs are given in Appendix B.

4. Numerical experiments

In this section, we will investigate numerical performances of our algorithms and compare them with those obtained by other methods. Numerical comparison will be conducted with reference to the following three aspects:

- Comparing performances of GDS, L-GDS and L-RGDS on box-constrained problems.
- Comparing performances of L-GDS, L-RGDS, the filled function methods in [40,41], HPSO in [42] and HABC (ABC in [27] + L-BFGS) on box-constrained problems.
- Comparing performances of L-GDS, L-RGDS and smoothing penalty methods in [43–45] for general constrained problems.

The box-constrained problems include two test sets composing of test problems from [22] and website (http://www.sfu.ca/ ssurjano/optimization.html). Appendix C shows the details of these test problems. The general constrained test problems are chosen from [43–46]. The codes are written in Matlab 7.5 with double precision arithmetic. The algorithms are carried out on a PC (Intel Core Duo 2.6 GHz, 1GB memory) with Windows 7 operation system.

For all test problems, we choose the initial matrix $H_0 = I$ (the identity matrix), $\zeta = 0.2$, $\eta = 0.25$, $\varepsilon_1 = \varepsilon_2 = \varepsilon_3 = \varepsilon_4 = 10^{-6}$ and the number of correction pairs $m = 5$ in L-BFGS, the parameter $r_0 = 0.3$ in Algorithm L-RGDS and the translation parameter $t = \frac{1}{3}$, $q_1 = 10$, $q_2 = 5$ and $N = 10$ in Algorithm GDS. We initialize L-BFGS with a random starting point in the search region and maintain the best iteration solution. During the execution of L-BFGS, numerical gradients are calculated using a two point computational formula. Thus, the number of function evaluations for each gradient calculation is equal to the dimension of the optimization problem.

4.1. Comparison between GDS, L-GDS and L-RGDS for box-constrained optimization problems

We first apply Algorithms GDS, L-GDS and L-RGDS to solve some box-constrained optimization problems. Since Algorithm GDS can be applied directly to solve these problems, $F_\sigma(z)$ is replaced by $f(x)$ in Algorithm GDS. For Test A8, the search routes generated by L-BFGS, GDS, L-GDS and L-RGDS from the same initial point are depicted in Fig. 3. Within the feasible region, this test problem has four local minima and two global minima. Fig. 3(a) shows that L-BFGS is trapped at a local minimizer. Fig. 3(b) shows that GDS has the capability to steer the search direction towards global minimizer. However, it suffers from slow local convergence. Fig. 3(c) and (d) shows that both L-GDS and L-RGDS can capture one of the global minimizers. However, the solution obtained by L-RGDS is more accurate. Since L-BFGS is a local search method, we will not present the numerical results obtained.

We depict a typical convergence trajectory for the test problems set B in a particular trial in Fig. 4. Here, the algorithm is said to have successfully captured a global solution in this trial if the error between the optimal objective function value and that at the solution obtained by the algorithm is smaller than the tolerance $\varepsilon = 10^{-6}$.

Table 1 shows that GDS, L-GDS and L-RGDS have the capability for global optimization. However, GDS suffers from slow convergence. Through incorporating L-BFGS to L-GDS and L-RGDS, their convergence rates are improved significantly. For example, for the test problem A2, GDS can only achieves the accuracy of 10^{-2}. After speed-up by L-BFGS, L-GDS can achieve the accuracy of 10^{-8} and L-RGDS can achieve the accuracy of 10^{-9}. Fig. 4 shows that GDS has the capability to help L-BFGS jump out from local basin. Taking B6 in Fig. 4 as an example, we can observe that after three runs of GDS, L-GDS captures a global solution.

To evaluate the numerical performances of GDS, L-GDS as well as L-RGDS for higher dimensional optimization problems, we apply GDS, L-GDS and L-RGDS to solve the test problems A4, A5 and A6 with three different dimensions 10, 50 and 100, respectively. All the experiments are run 50 times independently. From the numerical results, we can compare the performance of these algorithms in terms of the mean of the optimal objective function values obtained and the successful rate for finding a global minimizer. The numerical results are reported in Table 2. Table 2 clearly shows that for these test problems, GDS has the capability to steer the search towards global solution. However, its convergence is slow and it has difficulty for finding an accurate solution. After a certain number of iterations, for example, 1000 iterations, the median of the accuracy may still be under 10^{-2}. During the calculation of success rate, the algorithm is regarded as successful in a particular trial if the error between the optimal objective function value and that of the solution obtained by the algorithm is smaller than the tolerance $\varepsilon = 10^{-6}$. Comparing L-GDS and L-RGDS, we can clearly observe from Table 2 that L-GDS has a higher accuracy and a higher success rate than L-RGDS under the same number of executions of L-BFGS. In fact, the success rate depends heavily on the number k of executions of L-BFGS. The larger the k, the better the chance for L-GDS to capture a global solution. However, during the execution of L-RGDS, the search space is controlled by a random variable r and the parameter r_0. If $r < r_0$, the exploration search is skipped and the local search is continued from the previous iteration. This explains why L-RGDS has lower success rate but with less function evaluations. In applications, if an accurate solution is not required, L-RGDS is a good option. Table 2 shows the numbers of function evaluations for GDS, L-GDS and L-RGDS in which we can see that L-GDS requires the largest number of function evaluations, while GDS requires the least. Meanwhile, we can also observe that GDS has the worst success rate and L-GDS has the best success rate.

4.2. Comparison of L-GDS and L-RGDS with existing algorithms for box-constrained optimization problems

In this subsection, we will apply L-GDS and L-RGDS to solve some box-constrained optimization problems and compare their

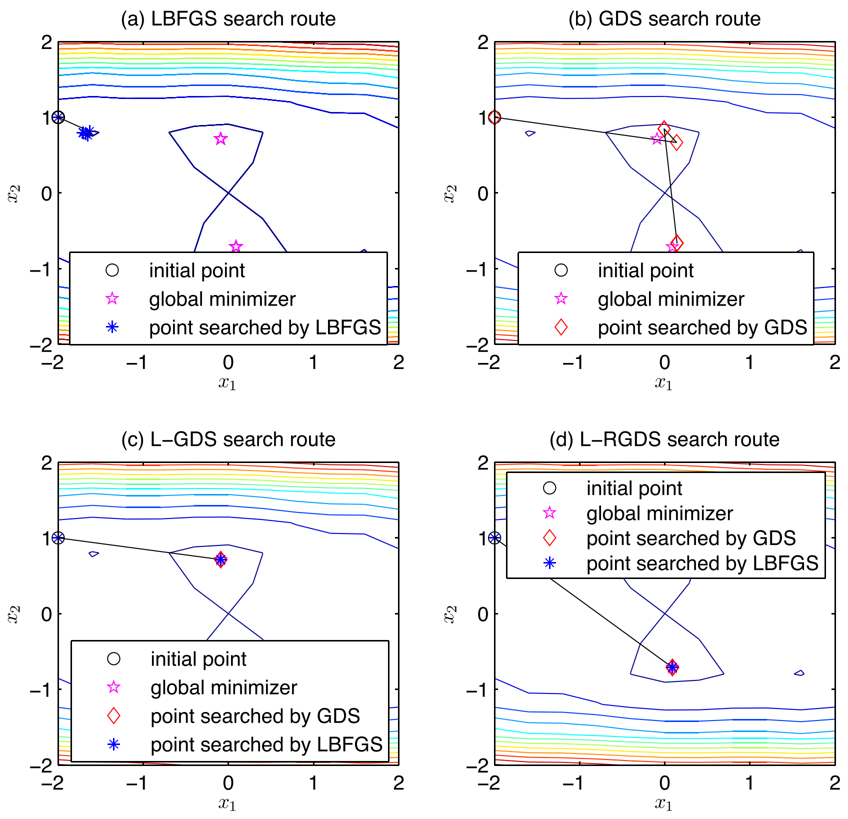

Fig. 3. Search route for Test A8 with 2-dimension within 20 iterations: (a) L-BFGS, (b) GDS, (c) L-GDS and (d) L-RGDS.

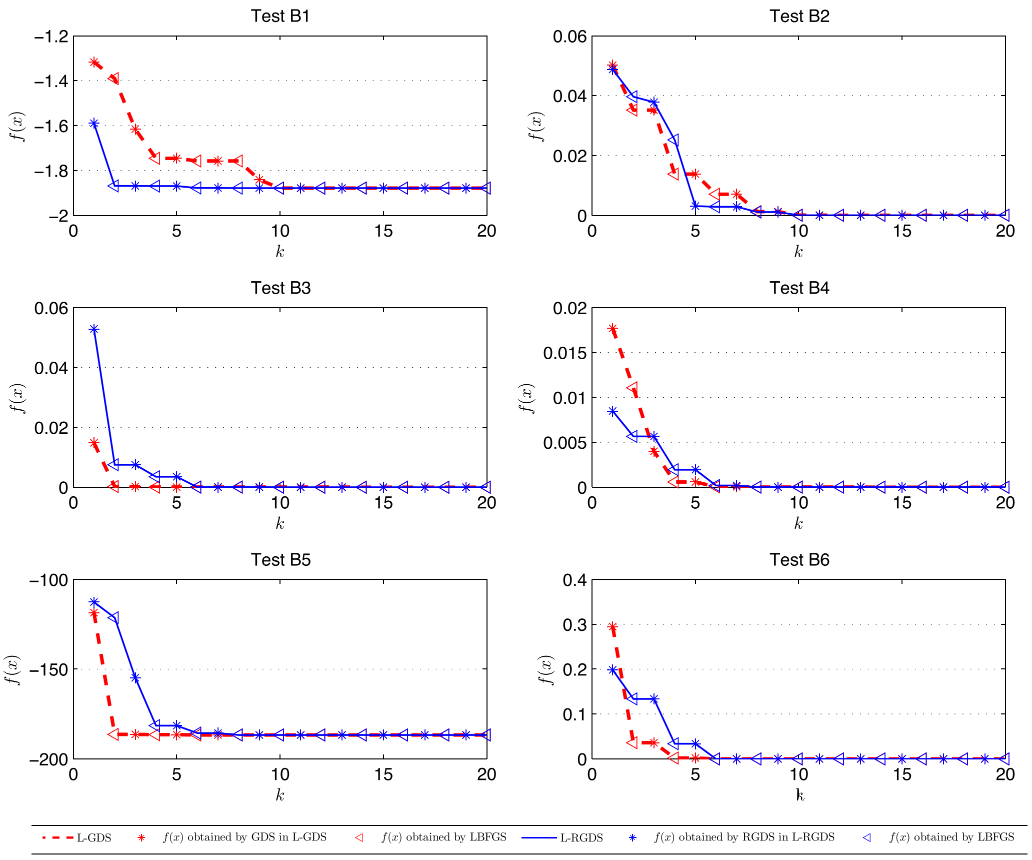

Fig. 4. Typical convergence trajectories generated by L-GDS and L-RGDS of the test problems from the test set B with 2-dimension.

Table 1
Numerical results for test problems with 2-dimension by GDS, L-GDS and L-RGDS for 50 independent runs. "Dim." indicates the dimension of test function, "Mean Eval." indicates the average number of evaluations of the function and its gradient.

Test	Dim.	Mean of $f(x^*)$			Mean Eval.		
		GDS	L-GDS	L-RGDS	GDS	L-GDS	L-RGDS
A1	2	3.2625e−04	2.8490e−33	7.8506e−24	100	215	110
A2		1.7221e−02	1.1780e−08	3.0057e−09	100	2132	907
A3		4.9756e+00	9.1551e−15	2.8263e−13	100	2114	1299
A4		1.4799e−02	1.5503e−16	3.3307e−16	100	1677	647
A5		2.3962e−02	2.8663e−34	2.7075e−24	100	2097	853
A6		3.0014e+00	3.0000e+00	3.0000e+00	100	2114	1516
A7		−1.0000e+00	−1.0000e+00	−1.0000e+00	100	205	205
A8		−1.0239e+00	−1.0316e+00	−1.0316e+00	100	2101	1489
B1		−1.9206e+00	−2.0000e+00	−1.9980e+00	310	3043	2905
B2		4.2560e−05	5.1563e−24	1.2513e−12	310	2375	1553
B3		3.9900e−04	7.5877e−22	1.6667e−22	310	1080	1030
B4		2.1300e−03	6.0737e−25	2.6402e−23	310	1082	1026
B5		−1.8275e+02	−1.8673e+02	−1.8673e+02	310	3242	2109
B6		2.3558e−31	2.3558e−31	2.3558e−17	310	217	207

Table 2
The performances of GDS, L-GDS and L-RGDS on 3 test problems with dimensions 10, 50 and 100. "Dim." indicates the dimension of test function, "Mean Eval." indicates the average number of evaluations of the function and its gradient.

Test	Dim.	Mean of $f(x^*)$			Mean Eval.			Success rate (%)		
		GDS	L-GDS	L-RGDS	GDS	L-GDS	L-RGDS	GDS	L-GDS	L-RGDS
A4	10	4.3832e−02	7.6106e−03	**7.4585e−03**	1000	30,841	26,281	15	93	92
A5		1.0376e−06	**1.6042e−29**	3.6082e−16	1000	20,843	8372	5	100	95
B6		10.5159	**3.110e−08**	3.1650e−02	1420	30,268	22,686	10	98	88
A4	50	1.0276e+00	**9.0262e−01**	9.5823e−01	2500	52,111	21,511	4	94	92
A5		1.2013e−07	**1.1547e−26**	1.2991e−19	5000	104,124	16,010	1	100	93
B6		2.0000e−00	**1.9970e−20**	5.808e−20	6010	60,168	39,934	3	88	85
A4	100	1.0960e+00	8.4023e−01	8.0670e−01	10,000	508,221	457,821	3	90	85
A5		2.4486e−08	**3.3930e−24**	4.3598e−21	10,000	508,224	203,613	1	100	97
B6		2.0000e−00	**3.1100e−04**	6.6222e−03	18,030	153,049	87,537	2	85	70

performance with some existing algorithms. Seven test problems chosen from the literature are solved by using our proposed hybrid algorithms. We then compare our results with those obtained by other existing methods which are the hybrid PSO algorithm (HPSO) in [42], hybrid original ABC with L-BFGS (denoted by HABC), and two filled function methods in [40,41]. In HPSO, simplex search and PSO are integrated together to solve global optimization problems. HABC is the integration of ABC in [27] with L-BFGS. The population size of ABC is set as 10 and the maximal iteration number is set as $q_2 = 10$ for inner cycle (i.e. ABC cycle) in HABC. For the filled function method, an auxiliary function, referred to as filled function, is defined by

$$P(x, x^*) = -\text{sign}(f(x) - f(x^*))\arctan(\|x - x^*\|),$$

where $\text{sign}(t) = \begin{cases} 1, & t \geq 0 \\ -1, & t < 0 \end{cases}$, and x^* is the current minimizer (i.e. local minimizer). Then, minimize $P(x, x^*)$ along several directions $d_j, j = 1, 2, \ldots$, from the iteration point x_k by the iterative formula

$$x_{l+1} = x_l + \lambda \frac{d_j}{\|d_j\|}$$

with step size λ and search direction d_j (see [40,41] for details). The corresponding numerical results are presented in Tables 3 and 5.

In addition, we apply statistical significance test, which is a meaningful way to study the difference between any two stochastic algorithms, to do comparison among five algorithms. Wilcoxon signed rank test is applied to determine the difference between paired scores. Based on the data from Tables 1–3, the statistical results based on the mean function values are presented in Table 4, where $R = R^-$ or R^+ is the sum of ranks based on the absolute value of the difference between two test algorithms. The sign of the difference between two independent samples is used to classify the two samples: the differences is above zero (positive rank R^+), or the difference is below zero (negative rank R^-).

Based on the statistical results in Tables 1–3, we introduce Wilcoxon Sign Rank with a statistical significance value $\alpha = 0.05$ to compare the statistical numerical performances of the algorithms. The null hypothesis is "there is no significant difference between the best or the mean values of the two samples". We use the well-known statistical software packages SPSS 19 to compute the p-value for these tests. Based on the test results/rankings, we assign one of the three signs (+, − and ≈), where the sign "+" (respectively, "−") means that the first algorithm is significantly better (respectively, worse) than the second algorithm and the sign ≈ means no significant difference between the two algorithms. The comparison results in Table 4 show that compared L-GDS with GDS, L-RGDS, HPSO and HABC, the null hypothesis is rejected. That is to say, the alternative hypothesis is accepted and the performance of L-GDS is better than the others.

Comparing with the hybrid method in [42], we can clearly observe from Table 3 that L-GDS and L-RGDS achieve higher accuracy for all the test problems. For the test problems A4 and A5, L-GDS and L-RGDS achieves better performances at the expense of the larger mean number of function and gradient evaluations. For the filled function methods in [40,41], all the methods have good chance to capture global solution. However, the computational complexity of the filled function methods is much higher than that of L-GDS or L-RGDS. For example, for the test problem B1, the number of function evaluations of the filled function method in [40] is 11,435. However, the numbers of function evaluations for L-GDS and L-RGDS are 1853 and 1074, respectively. This indicates

Table 3
Numerical results of HPSO, L-GDS and L-RGDS. "Dim." indicates the dimension of test function, "Mean Eval." indicates the average number of evaluations of the function and its gradient, "Mean Err." indicates the average error between the best successful point achieved and the known global optimum.

Test	Dim.	Mean Eval.				Mean Err.			
		HPSO [42]	HABC	L-GDS	L-RGDS	HPSO [42]	HABC	L-GDS	L-RGDS
A1	3	291	346	171	132	5.00e−05	2.3100e−06	**3.2736e−23**	1.6540e−22
A2	2	339	392	208	101	3.00e−05	**1.6742e−05**	3.6000e−05	1.7500e−04
A3	10	NA	392	301	164	NA	3.6442e−08	1.4921e−13	**0.000e+00**
A4	8	1354	2333	3235	1321	4.10e−04	3.0120e−04	**2.2386e−06**	1.9230e−05
A5	5	1394	2392	1433	2421	2.60e−05	1.3512e−05	**5.3736e−11**	1.5839e−06
A6	2	217	1476	1121	807	3.00e−05	5.5233e−06	**2.5373e−08**	6.0042e−07
A7	2	165	213	178	137	4.00e−05	8.5380e−06	**3.5083e−09**	4.7296e−06
A8	2	151	181	183	145	4.00e−05	3.0000e−05	**3.5083e−07**	4.7296e−05
A9	2	165	213	253	199	4.00e−05	8.5380e−06	**2.8420e−07**	4.7296e−06
A10($H_{3,4}$)	3	271	213	432	354	2.40e−05	8.5380e−06	**2.1335e−07**	2.0016e−06
A10($H_{6,4}$)	6	1541	213	1736	1576	2.50e−03	8.5380e−03	**2.4177e−04**	6.4376e−04
A11($S_{4,5}$)	4	1177	213	1654	1312	2.00e−04	8.5380e−05	**3.2097e−07**	7.4716e−05
A11($S_{4,7}$)	4	1130	213	1020	936	1.70e−04	8.5380e−06	**4.7716e−05**	4.2617e−04
A11($S_{4,10}$)	4	1179	213	1874	1293	1.50e−04	8.5380e−05	**1.9212e−05**	7.9296e−05

Table 4
Wilcoxon sign rank results on mean error of optimum values obtained by GDS, L-GDS, L-RGDS, HPSO and HABC, where Wilcoxon sign rank test for L-GDS vs GDS, L-RGDS vs GDS is based on Tables 1 and 2, and the others are based on Table 3.

Algorithm	R^-	R^+	p value	Decision	Algorithm	R^-	R^+	p value	Decision
L-GDS to GDS	78	0	0.002	+	L-RGDS to GDS	90	1	0.002	+
L-GDS to L-RGDS	103	2	0.002	+					
L-GDS to HPSO	90	1	0.002	+	L-RGDS to HPSO	69	22	0.100	–
L-GDS to HABC	84	21	0.048	+	L-RGDS to HABC	39	66	0.397	–

Table 5
Numerical performances of filled function methods, L-GDS and L-RGDS. "Dim." indicates the dimension of test function, "Mean Eval." indicates the average number of evaluations of the function and its gradient. "NA" indicates that this method is not applicable to solve this problem. "Mean Eval." for L-GDS and L-RGDS are the average of 50 independent runs.

Test	Dim	Mean Eval.			
		Filled function [40]	Filled function [41]	L-GDS	L-RGDS
B1	2	11,435	6579	1853.2	1074.6
B2	2	1269	7651	2204.3	969.7
B3	2	14,269	NA	2138.8	1074.1
A8	2	3572	3527	2204.5	1186.1
B4	2	3113	3303	2177.6	1130.5
B5	2	8499	60,577	2241.2	1064.5
B6	2	87,115	NA	2204.5	1105.3
B6	3	136,026	NA	3253.5	1571.5
B6	5	347,184	155,820	5016.7	2942.7
B6	7	246,973	NA	5161.7	2939.9
B6	10	309,379	861,652	49,946	34,870

a reduction of 83% on the number of function evaluations for L-GDS, while the reduction for L-RGDS is about 90%. However, we should point that in every execution, each of the filled function methods has captured a global solution, while L-GDS and L-RGDS can only capture global solution in the sense of probability one.

4.3. *Integrating EPM and L-GDS or L-RGDS for general constrained optimization problems*

We will evaluate the numerical performance of the two algorithms through solving constrained optimization problems. Let us consider four optimization problems listed in Appendix D. These optimization problems have been studied in some existing literature. During the implementation process, we set $\alpha = 2$, $\beta = 2$ and $\bar{\epsilon} = 1$. All the results for each problem presented in Tables 6–9 are the best results achieved by L-GDS and L-RGDS among 5 independent trials.

The smoothing l_1 exact penalty function method (SOPFA) proposed in [43] is used to solve Example 1 in Appendix D. The main idea of SOPFA is to transform Problem (P) into the minimization of a continuously differentiable function on $\mathbb{R}^n$ defined by:

$$F(x, \sigma, \epsilon) = f(x) + \sigma \sum_{i=1}^{l+m} p_\epsilon(g_i(x)),$$

where $p_\epsilon(t) = \begin{cases} \dfrac{3}{2}\epsilon e^{\frac{t}{\epsilon}} - 2\epsilon, & t \le 0 \\ t - \dfrac{1}{2}\epsilon e^{-\frac{t}{\epsilon}}, & t > 0 \end{cases}$ for an $\epsilon > 0$, and a $\sigma > 0$.

Next, choose an initial point x_0. Set $\epsilon_0 > 0$, $\sigma_0 > 0$, $0 < \eta < 1$, and $\varsigma > 1$. Then, x_k is used as the starting point to minimize $F(x, \sigma_k, \epsilon_k)$ by a quasi-Newton method. After which, let $\sigma_{k+1} = \varsigma\sigma_k$ and $\epsilon_{k+1} = \eta\epsilon_k$. The process is repeated until no further improvement is achieved.

Let the initial point x_0 be generated randomly in the feasible region Ω, $\sigma_0 = 10$, $\sigma_{\max} = 10^2$ and $\varsigma = 3$ in our Algorithm 1. Numerical results obtained by L-GDS and L-RGDS are presented in Table 6. This table shows that the solution obtained by L-GDS is better than

Table 6
Numerical results obtained by L-GDS, L-RGDS and SOPFA for Example 1.

Algorithm	σ	ϵ	$f(x^*)$	$g(x^*)$	x^*
L-GDS	30	8.265412e−04	**1.837615**	(−0.774352, −0.000001)	(0.724702, 0.399076)
L-RGDS	30	5.802941e−05	1.837749	(−0.777405, −0.000077)	(0.726019, 0.399458)
SOPFA [43]	90		1.837623	(−0.7760112, −0.000044)	(0.7254170, 0.3992834)

Table 7
Numerical results obtained by L-GDS, L-RGDS, Algorithm I in [44] and Algorithm I in [45] for Example 2, where best results are presented for L-GDS and L-RGDS over 5 independent runs.

Algorithm	σ	ϵ	$f(x^*)$	$g(x^*)$
L-GDS	10	1.136400e−02	−44.206878	(−5.478152e−03, −1.765023e−03, −1.682682e+00)
	100	1.001994e−08	**−44.213915**	(−7.897230e−03, −1.151412e−04, −2.107105e+00)
L-RGDS	10	9.836367e−02	−44.223817	(−3.545977e−03, 3.962968e−03, −1.887517e+00)
	100	1.000000e−08	−44.221052	(−8.360936e−06, −2.346339e−03, −1.994742e+00)
	1000	4.381713e−03	**−44.208249**	(−1.440614e−03, −1.625671e−03, −2.062844e+00)
Alg. I in [44]	10		−44.455547	(3.764288e−02, 9.8054105e−02, −1.773709e+00)
	100		−44.256315	(3.743516e−03, 9.9164819e−03, −1.871979e+00)
Alg. I in [45]	10		−44.547342	(2.368187e−02, 1.505667e−01, −1.706287e+00)
	100		−44.237119	(2.219450e−04, 1.569493e−03, −1.880785e+00)
	1000		−44.233877	(−3.236100e−07, 2.029069e−05, −1.883000e+00)

Table 8
Results of L-GDS, L-RGDS and other algorithm in literatures for Example 3.

Algorithm	σ	ϵ	$f(x^*)$	$g(x^*)$
L-GDS	10	8.242863e−03	944.251600	(9.264565e−05, 1.135775e−04, −5.486185e−01)
	100	1.496411e−02	944.238079	(7.135391e−03, 3.362558e−04, −1.280594e+00)
	1000	4.740569e−03	944.234918	(9.000140e−04, 3.100654e−04, −2.819471e+00)
	10,000	2.971434e−03	**944.219625**	(5.073684e−04, 2.475933e−04, −1.371761e+00)
L-RGDS	10	4.899230e−03	945.705218	(7.009727e−04, 1.724624e−03, −8.497550e+00)
	100	6.309856e−03	944.211001	(3.198315e−03, 1.475358e−03, −2.167139e+00)
	1000	7.974351e−03	944.158110	(3.169335e−02, 1.589298e−02, −1.244968e+00)
	10,000	2.203397e−02	**944.002748**	(1.367182e−01, 3.290986e−02, −2.841818e−02)
Alg. II in [44]	10		944.980301	(1.082548e−01, 6.634798e−03, −1.945075e+00)
	100		944.192098	(1.083098e−02, 6.509816e−04, −1.866579e+00)

Table 9
Results of L-GDS, L-RGDS and other algorithm in literatures for Example 4.

Algorithm	σ	ϵ	$f(x^*)$	$g(x^*)$
L-GDS	100	3.835429e−02	123.955332	(1.3210e−02, 1.1490e−02, 2.1951e−02, −6.5167e−03, −6.2642e+00)
	1000	7.871252e−03	124.200553	(1.0136e−02, 3.3292e−04, 1.0962e−03, −5.2929e−02, −6.9748e+00)
	10,000	2.261166e−02	**122.79290**	(6.1642e−02, 6.9071e−02, 2.6990e−02, −4.0147e−02, −6.6558e+00)
L-RGDS	80	8.559575e−03	124.039196	(4.5024e−05, 8.4472e−04, 2.5680e−03, −2.3834e−02, −6.7553e+00)
	100	3.342463e−02	123.964508	(7.7168e−03, 1.6201e−02, 3.9011e−04, 2.4601e−03, −5.7714e+00)
	1000	4.995991e−02	**123.98075**	(2.8058e−02, 1.4849e−02, 5.2620e−02, 2.5179e−02, −1.9501e+00)
Alg. in [46]	80	1.75e−03	123.99	(−1.0000e−10, −1.0000e−10, −1.0000e−10, 3.0000e−03, −8.7690e+00)

that obtained by SOPFA. However, for only one run of L-RGDS, it cannot obtain a better solution than that in [43]. This is partly caused by the random nature in the running of GDS in L-RGDS. If we increase to 5 runs, L-RGDS can capture a solution better than that in [43] in most of the trials.

For Example 2 considered in [44,45], we set $\sigma_0 = 10$ and $\varsigma = 10$. The numerical results obtained by L-GDS, L-RGDS as well as Algorithm I in [44] and Algorithm I in [45] are shown in Table 7. From this table, we can observe that both Algorithm I in [44] and Algorithm I in [45] cannot find a feasible solution even the penalty parameter is increased to 1000. On the other hand, all solutions obtained by L-GDS and L-RGDS are feasible even when the penalty parameter is only set as $\sigma = 10$.

We consider Example 3 in [44] and Example 4 in [46] by using the same values of the parameters for those given in Example 2. The numerical results obtained are reported in Table 7 and 8, respectively. From the four tables above, we can clearly observe that our algorithm, L-GDS and L-RGDS, achieve more accurate solutions than those obtained in the existing literature in terms of satisfying feasibility tolerance of the constraints.

5. Conclusion

Deterministic optimization methods, such as BFGS method, are known for their fast convergence for solving convex optimization problems. However, they tend to be trapped in local minima for non-convex problems. In this paper, we proposed two hybrid algorithms for constrained global optimization. Based on the exact penalty function method, the constrained optimization problems were transformed into box-constrained optimization problems. Then, a novel reposition technique, *Greedy Diffusion Search* (GDS), is proposed to integrate with limited memory BFGS (L-BFGS) under two different strategies, where GDS is to enable the algorithm to escape from local minima. We have shown that our algorithms are convergent to a global minimizer with probability one. 18 box constrained optimization problems and 4 general constrained optimization problems are solved by the proposed methods. The results obtained were compared with those obtained by existing methods. Comparisons show that our methods can achieve higher accuracy with less number of function evaluations. However, as L-BFGS is used in our two hybrid methods, the functions involved in

Problem (P) are required to be smooth. It is clearly an important task to develop effective hybrid algorithms to solve non-smooth optimization problems.

Appendix A. Two theorems for Section 2

Theorem 1. *Let (x^*, ϵ^*) be a solution of Problem (P_σ). Then x^* is a solution of Problem (P) if and only if $\epsilon^* = 0$.*

Proof. The proof is similar to that given for Lemma 3 in [31]. □

Theorem 2. *Suppose that Mangasarian-Fromovitz constraint qualification is satisfied at every solution x^* of Problem (P). Let $\{\sigma_k\}_{k=1}^\infty$ be an increasing sequence of penalty parameters such that $\sigma_k \to \infty$ as $k \to \infty$. Furthermore, let $(x^{k,*}, \epsilon^{k,*})$ be a local solution of Problem (P_{σ_k}). Suppose that $\{F_{\sigma_k}(x^{k,*}, \epsilon^{k,*})\}_{k=1}^\infty$ is bounded. Then, for all sufficiently large k, $x^{k,*}$ is also a local solution of Problem (P).*

Proof. The proof is similar to that given for Theorem 4 in [31].□

Appendix B. Theorems for Section 3.4

In this appendix we will establish the convergence of Algorithm L-GDS and Algorithm L-RGDS. To achieve this task, we first need to establish the convergence of Algorithm GDS with probability one. More specifically, we will show that the sequence $z^{(k)}$ obtained by either L-GDS or L-RGDS converges with probability one to the region Ω_ε for any given $\varepsilon > 0$, where $\Omega_\varepsilon = \{z \in \Omega : |F_\sigma(z) - F_\sigma(z^*)| \leq \varepsilon\}$.

To proceed further, we require the following assumptions:

- **A1:** The function $F_\sigma(z)$ is continuously differentiable with respect to z for any given $\sigma > 0$.
- **A2:** $\min_{z\in\Omega} F_\sigma(z) > -\infty$.

Let $z^{(k)}$, $k = 1, 2, \ldots$, be the sequence generated by Algorithm GDS. By virtue of the greedy rule given by (17), Algorithm GDS is descent. In addition, $\min_{z\in\Omega} F_\sigma(z) > -\infty$ by Assumption A2, $\lim_{k\to\infty} F_\sigma(z^{(k)})$ exists. Now the convergence with probability one is defined formally in the following definition.

Definition 1. Let $z^{(k)}$, $k = 1, 2, \ldots$, be the minimizing sequence obtained by Algorithm GDS. Suppose that z^* is one of the global optimal solutions of Problem (P_σ). If

$$\lim_{k\to\infty} P\left(F_\sigma(z^{(k)}) = F_\sigma(z^*)\right) = 1, \tag{18}$$

then the algorithm is said to converge with probability one to a global minimum.

Lemma 1. *Let $z^{(k)}$ be generated by Algorithm GDS and let z^* be one of the global optimal solutions of Problem (P_σ). Then, (18) holds if and only if for any given $\varepsilon > 0$,*

$$\lim_{k\to\infty} P\left(\rho(z^{(k)}, \Omega_\varepsilon) \geq \delta\right) = 0, \quad \text{foranygiven}\, \delta > 0, \tag{19}$$

where $\rho(z^{(k)}, \Omega_\varepsilon) = \inf_{z\in\Omega_\varepsilon} \|z - z^{(k)}\|$.

Proof. Note that

$$F_\sigma(z^{(k)}) \geq F_\sigma(z^{(k+1)}) \geq F_\sigma(z^*),$$

and hence

$$\rho(z^{(k)}, \Omega_\varepsilon) \geq \rho(z^{(k+1)}, \Omega_\varepsilon) \geq 0.$$

Thus, $\lim_{k\to\infty} \rho(z^{(k)}, \Omega_\varepsilon)$ exists. Note that

$$\lim_{k\to\infty} F_\sigma(z^{(k)}) = F_\sigma(z^*) \,\text{ifandonlyif}\, \lim_{k\to\infty} \rho(z^{(k)}, \Omega_\varepsilon) = 0, \,\text{forany}\, \varepsilon > 0.$$

The result follows readily. □

For Algorithm GDS, we have the following convergence theorem.

Theorem 3. *Algorithm GDS converges with probability one to a global minimum.*

Proof. We will adapt the proof given for the main theorem in [47] to prove this theorem. In light of Definition 1 and Lemma 1, we only need to prove that for any given $\varepsilon > 0$ and $\delta > 0$, $\lim_{l\to\infty} P\left(\rho(z^{(l)}, \Omega_\varepsilon) \geq \delta\right) = 0$.

By virtue of Assumption (A1), $F_\sigma(z)$ is continuously differentiable in the box set Ω. Thus, $F_\sigma(z)$ is uniformly continuous. This means that there exists a $\widetilde{\delta}$, such that for any $\bar{z} \in \Omega$, if z satisfies $\|z - \bar{z}\| \leq \widetilde{\delta}$, we have $|F_\sigma(z) - F_\sigma(\bar{z})| \leq \varepsilon/2$. In particular, if z satisfies $\|z - z^*\| \leq \widetilde{\delta}$, we have $0 \leq F_\sigma(z) - F_\sigma(z^*) \leq \varepsilon/2$. Define the ball $B_{\widetilde{\delta}}(z^*) = \{z \in \Omega : \|z - z^*\| \leq \widetilde{\delta}\}$. If $z^{(l)} \notin B_{\widetilde{\delta}}(z^*)$, $z^{(l)} \in \Omega$ and $z^{(l+1)} \in B_{\widetilde{\delta}}(z^*)$, then there exists a $\bar{z} \in \bigcup_{i=1}^{q_1} \widetilde{z}_i^{(l+1)} \bigcup_{j=1}^{q_2} \hat{z}_j^{(l+1)}$ such that $\bar{z} \in B_{\widetilde{\delta}}(z^*)$, where $\widetilde{z}_i^{(l+1)}$, $i = 1, \ldots,$ and $\hat{z}_j^{(l+1)}$, $j = 1, \ldots, q_2$, are defined in Algorithm GDS. Note that $\widetilde{z}_i^{(l+1)} \in \Omega$, $i = 1, \ldots,$ and $\hat{z}_j^{(l+1)} \in \Omega$, $j = 1, \ldots, q_2$, are generated randomly with the uniform distribution in Ω, we have

$$\begin{aligned} & P\left(z^{(l+1)} \in B_{\widetilde{\delta}}(z^*) | z^{(l)} \notin B_{\widetilde{\delta}}(z^*), z^{(l)} \in \Omega\right) \\ \geq\ & P\left(\hat{z}_1^{(l+1)} \in B_{\widetilde{\delta}}(z^*) | z^{(l)} \notin B_{\widetilde{\delta}}(z^*), z^{(l)} \in \Omega\right) \\ =\ & P\left(\hat{z}_1^{(l+1)} \in B_{\widetilde{\delta}}(z^*)\right) = \frac{m\left(B_{\widetilde{\delta}}(z^*)\right)}{m(\Omega)}, \end{aligned} \tag{20}$$

where $m\left(B_{\widetilde{\delta}}(z^*)\right)$ is the volume of the set $B_{\widetilde{\delta}}(z^*)$ and $m(\Omega)$ is the volume of the set Ω. The last inequality in (20) is valid due to the independency of $\hat{z}_1^{(l+1)}$ and $z^{(l)}$. Through choosing $\widetilde{\delta}$ appropriately, we have $\frac{m\left(B_{\widetilde{\delta}}(z^*)\right)}{m(\Omega)} < 1$. For notational brevity, let $0 < \gamma = \frac{m\left(B_{\widetilde{\delta}}(z^*)\right)}{m(\Omega)} < 1$.

Now we introduce the auxiliary variable $y^{(l)}$ defined by:

$$y^{(l)} = \begin{cases} 1, & \text{if} \quad F_\sigma(z^{(l)}) - F_\sigma(z^{(l-1)}) \geq \varepsilon/2; \\ 0, & \text{otherwise}. \end{cases} \tag{21}$$

Denote $K = \lfloor 2(F_\sigma(z(0)) - F_\sigma(z^*))/\epsilon \rfloor + 1$, where $\lfloor \cdot \rfloor$ is the floor function. Through direct verification, we can show that $z^{(l)} \in \Omega_\epsilon$ if $\sum_{i=1}^{l} y^{(i)} \geq K$. Equivalently, we have $\sum_{i=1}^{l} y^{(i)} < K$ if $z^{(l)} \notin \Omega_\epsilon$.

Note that if $\hat{z}_1(l) \in B_{\widetilde{\delta}}(z^*)$, we have $F_\sigma(\hat{z}_1^{(l)}) - F_\sigma(z*) \leq \varepsilon/2$. In addition, it follows from $z^{(l-1)} \notin \Omega_\epsilon$ that $F_\sigma(\hat{z}_1^{(l-1)}) - F_\sigma(z*) \geq \varepsilon$. Thus, we have

$$\begin{aligned} & P\left(y^{(l)} = 1 | z^{(l-1)} \in \Omega, z^{(l-1)} \notin \Omega_\epsilon\right) \\ =\ & P\left(F_\sigma(z^{(l)}) - F_\sigma(z^{(l-1)}) \geq \varepsilon/2 \;\middle|\; z^{(l-1)} \in \Omega, z^{(l-1)} \notin \Omega_\epsilon\right) \\ \geq\ & P\left(\hat{z}_1^{(l)} \in B_{\widetilde{\delta}}(z^*)\right) = \gamma. \end{aligned}$$

The above inequality is equivalent to

$$P\left(y^{(l)} = 0 | z^{(l-1)} \in \Omega, z^{(l-1)} \notin \Omega_\epsilon\right) \leq 1 - \gamma. \tag{22}$$

For any $\delta > 0$,

$$\begin{aligned} P\left(\rho(z^{(l)}, \Omega_\varepsilon) \geq \delta\right) &= P\left(\rho(z^{(l)}, \Omega_\varepsilon) \geq \delta \ \mid \ z^{(i)} \in \Omega, z^{(i)} \notin \Omega_\varepsilon, i=1,\ldots,l-1\right) \\ &\leq P\left(z^{(l)} \notin \Omega_\epsilon \ \mid \ z^{(i)} \in \Omega, z^{(i)} \notin \Omega_\varepsilon, i=1,\ldots,l-1\right) \\ &\leq P\left(\sum_{i=1}^{l} y^{(i)} < K \ \mid \ z^{(i)} \in \Omega, z^{(i)} \notin \Omega_\varepsilon, i=1,\ldots,l-1\right). \end{aligned} \tag{23}$$

Similar to the proof given in [47], we can show that

$$\lim_{l\to\infty} P\left(\sum_{i=1}^{l} y^{(i)} < K \ \mid \ z^{(i)} \in \Omega, z^{(i)} \notin \Omega_\varepsilon, i=1,\ldots,l-1\right)$$
$$\leq \lim_{l\to\infty} \sum_{j=0}^{K-1} C_l^j (1-\gamma)^{k-j} = 0. \tag{24}$$

where $C_l^j = \frac{j!}{l!(l-j)!}$. Substituting (24) into (23), we obtain for any given $\varepsilon > 0$ and $\delta > 0$, $\lim_{l\to\infty} P\left(\rho(z^{(l)}, \Omega_\varepsilon) \geq \delta\right) = 0$. □

For Algorithm L-GDS and Algorithm L-RGDS, we have the following convergence theorem.

Theorem 4. *Algorithm L-GDS (respectively, Algorithm L-RGDS) converges with probability one to a global minimum.*

Proof. The results follow readily from the local convergence of Algorithm L-BFGS and Theorem 3. □

Appendix C. List of test functions for box constrained optimization

Separability: A function of p variables is called separable, if it can written as a sum of p functions of just one variable [48]. Otherwise, a function is called nonseparable. In general, separable functions are relatively easy to solve, when compared with nonseparable (for short, **Insep.**) functions.

A 1. *De Joung Function (n variables)*
$DJ_n = \sum_{i=1}^{n} x_i^2$;
search domain: $-100 \leq x_i \leq 100$, $i = 1, 2, \ldots, n$;
no local minimum;
1 global minimum: $x^* = (0, \ldots, 0)$, $DG_n(x^*) = 0$;
separable.

A 2. *Rosenbrock (n variables)*
$R_n = \sum_{i=1}^{n-1}\left[100(x_{i+1} - x_i^2)^2 + (1-x_i)^2\right]$;
search domain: $-30 \leq x_i \leq 30$, $i = 1, 2, \ldots, n$;;
no local minimum;
1 global minima, $x^* = (1, \ldots, 1)$; $R_n(x^*) = 0$;
nonseparable.

A 3. *Rastrigin (n variables)*
$RT_n(x) = 10n + \sum_{i=1}^{n}\left[x_i^2 - 10\cos(2\pi x_i)\right]$
search domain: $-5.12 \leq x_i \leq 5.12$, $i = 1, 2, \ldots, n$;
several local minima;
1 global minimum, $x^* = (0, \ldots, 0)$, $RT_n(x^*) = 0$
separable.

A 4. *Griewank (n variables)*
$GR(x) = 1 + \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos\frac{x_i}{\sqrt{i}}$
search domain: $-100 \leq x_i \leq 100$, $i = 1, 2, \ldots, n$;
several local minima;
1 global minimum, $x^* = (0, \ldots, 0)$, $GR(x^*) = 0$
nonseparable.

A 5. *Zakharov (n variables)*
$Z_n(x) = \sum_{i=1}^{n} x_i^2 + (\frac{1}{2}\sum_{i=1}^{n} ix_i)^2 + (\frac{1}{2}\sum_{i=1}^{n} ix_i)^4$
search domain: $-5 \leq x_i \leq 10$, $i = 1, 2, \ldots, n$;
several local minima;
1 global minimum, $x^* = (0, \ldots, 0)$, $Z_n(x^*) = 0$;
nonseparable.

A 6. *Goldstein-Price (2 variables)*
$GP(x) = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)]$
$[30 + (2x_1 - 3x_2)^2\left(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2\right)]$;
search domain: $-2 \leq x_i \leq 2$, $i = 1, 2$;
several local minima;
1 global minimum, $x^* = (0, -1)$, $GP(x^*) = 3$;
nonseparable.

A 7. *Easom (2 variables)*
$ES(x) = -\cos(x_1)\cos(x_2)e^{-(x_1-\pi)^2-(x_2-\pi)^2}$;
search domain: $-100 \leq x_i \leq 100$, $i = 1, 2, \ldots, n$;
several local minima;
1 global minimum, $x^* = (\pi, \pi)$, $ES(x^*) = -1$;
separable.

A 8. *Six-Hump (2 variables)*
$SH(x) = \left(4 - 2.1x_1^2 + \frac{x_1^4}{3}\right)x_1^2 + x_1x_2 + \left(4x_2^2 - 4\right)x_2^2$;
search domain: $-5 \leq x_i \leq 5$, $i = 1, 2$;
several local minima;
2 global minima, $x^* = (\pm 0.0898, \mp 0.7126)$, $SH(x^*) = -1.0316$;
nonseparable.

A 9. *Branin RCOS (2 variables)*
$RC(x) = (x_2 - (5.1/4\pi^2)x_1^2 + (5/\pi)x_1 - 6)^2 + 10(1 - (1/8\pi))\cos(x_1) + 10$;
Range of initial points: $-5 < x_1 < 10$, $0 < x_2 < 15$;
3 global minima: $(x^*) = (-\pi, 12.275)$, $(\pi, 2.275)$, $(9.42478, 2.475)$; $RC(x^*) = 0.397887$;
separable.

A 10. *Hartmann Function ($H_{3,4}$) (3 variables)*
$H_{3,4}(x) = -\sum_{i=1}^{4} c_i e^{-\sum_{j=1}^{3} a_{ij}(x_j - p_{ij})^2}$;
Range of initial points: $0 < x_j < 1$, $j = 1, 2, 3$;
4 local minima;
1 Global minimum: $x^* = (0.114614, 0.555649, 0.852547)$, $H_{3,4}(x^*) = -3.86278$;
nonseparable.

i	a_{ij}			c_i	p_{ij}		
1	3.0	10.0	30.0	1.0	0.689	0.1170	0.2673
2	0.1	10.0	35.0	1.2	0.4699	0.4387	0.7470
3	3.0	10.0	30.0	3.0	0.1091	0.8732	0.5547
4	0.1	10.0	35.0	3.2	0.0381	0.5743	0.8828

A 11. *Hartmann Function ($H_{6,4}$, 6 variables)*
$H_{6,4}(x) = -\sum_{i=1}^{4} c_i e^{-\sum_{j=1}^{6} a_{ij}(x_j - p_{ij})^2}$;
Range of initial points: $0 < x_j < 1$, $j = 1, \ldots, 6$;
6 local minima;
1 global minimum: $x^* = (0.201690, 0.150011, 0.476874, 0.275332, 0.311652, 0.657300)$, $H_{6,4}(x^*) = -3.32237$;
nonseparable.

i	a_{ij}						c_i	p_{ij}					
1	10.0	3.0	17.0	3.50	1.70	8.00	1.0	0.1312	0.1696	0.5569	0.0124	0.8283	0.5886
2	0.05	10.0	17.0	0.10	8.00	14.00	1.2	0.2329	0.4135	0.8307	0.3736	0.1004	0.9991
3	3.00	3.50	1.70	10.0	17.00	8.00	3.0	0.2348	0.1451	0.3522	0.2883	0.3047	0.6650
4	17.00	8.00	0.05	10.00	0.10	14.00	3.2	0.4047	0.8828	0.8732	0.5743	0.1091	0.0381

A 12. *Shekel Functions ($S_{4,m}$, 4 variables)*
$S_{4,m}(x) = -\sum_{i=1}^{m}(\sum_{j=1}^{4}(x_j - a_{ij})^2 + c(i))^{-1}$;
3 functions are considered, namely: $S_{4,5}$, $S_{4,7}$ and $S_{4,10}$;
Range of initial points: $0 < x_j < 10$, $j = 1, \ldots, 4$;
m local minima;
1 global minimum: $x^* = (4, 4, 4, 4)$, $S_{4,5}(x^*) = -10.1532$, $S_{4,7}(x^*) = -10.4029$ and $S_{4,10}(x^*) = -10.5364$;
nonseparable.

i	a_{ij}				c_i
1	4.0	4.0	4.0	4.0	0.1
2	1.0	1.0	1.0	1.0	0.2
3	8.0	8.0	8.0	8.0	0.2
4	6.0	6.0	6.0	6.0	0.4
5	3.0	7.0	3.0	7.0	0.4
6	2.0	9.0	2.0	9.0	0.6
7	5.0	5.0	3.0	3.0	0.3
8	8.0	1.0	8.0	1.0	0.7
9	6.0	2.0	6.0	2.0	0.5
10	7.0	3.6	7.0	3.6	0.5

B 1. *Rastrigin (2 variables)*
$RT(x) = x_1^2 + x_2^2 - \cos(18x_1) - \cos(18x_2)$;
search domain: $-1 \le x_i, x_2 \le 1$;
several local minima;
1 global minimum: $x^* = (0, 0)$, $RT(x^*) = -2$;
separable.

B 2. *2-D Function (2 variables)*
$f(x) = [(1 - 2x_2 + c\sin(4\pi x_2) - x_1]^2 + [x_2 - 0.5\sin(2\pi x_1)]^2$, *where* $c = 0.2, 0.5$ *and* 0.05 .;
search domain: $0 \le x_1 \le 10$, $-10 \le x_2 \le 0$;
no local minimum;
1 global minima, $x^* = (1.8784, -0.3459)$, $(1.0000, 0.0000)$ and $(1.5975, -0.2874)$ for $c = 0.2, 0.5$ and 0.05, respectively. $f(x^*) = 0$;
nonseparable.

B 3. *Three-Hump Camel Back (2 variables)*
$CB(x) = 2x_1^2 - 1.05x_1^4 + \frac{1}{6}x_1^6 - x_1x_2 + x_2^2$;
search domain: $-3 \le x_1, x_2 \le 3$;
several local minima;
1 global minimum,$x^* = (0, 0)$, $CB(x^*) = 0$
separable.

B 4. *Treccani (2 variables)*
$f(x) = x_1^4 + 4x_1^3 + 4x_1^2 + x_2^2$;
search domain: $-3 \le x_1, x_2 \le 3$;
several local minima;
1 global minimum, $x^* = (-2.0, 0.0)$, $f(x^*) = 0$
separable.

B 5. *2-D Shubert (2 variables)*
$SH(x) = (\sum_{i=1}^{5} i\cos[(i+1)x_1 + i])(\sum_{i=1}^{5} i\cos[(i+1)x_2 + i])$
search domain: $-10 \le x_1, x_2 \le 10$;
several local minima;
1 global minimum,$x^* = (5.4829, 4.8581)$, $SH(x^*) = -186.7309$
nonseparable.

B 6. *Levy (n variables)*
$LV(x) = \frac{\pi}{n}\{10\sin^2(\pi x_1) + (x_n - 1)^2 + \sum_{i=1}^{n-1}\left[(x_i - 1)^2(1 + 10\sin^2(\pi x_{i+1}))\right]\}$
search domain: $-1.0 \le x_i \le 1.0$, $i = 1, 2, \ldots, n$;
several local minima;
1 global minimum,$x^* = (1, 1, \ldots, 1)$, $LV(x^*) = 0$
nonseparable.

Appendix D. Four test examples for general constrained optimization

Example 1.

$$\begin{aligned} \min \quad f(x) &= x_1^2 + x_2^2 - \cos(17x_1) - \cos(17x_2) + 3 \\ \text{s.t.} \quad g_1(x) &= (x_1-2)^2 + x_2^2 - 1.6^2 \le 0 \\ g_2(x) &= x_1^2 + (x_2-3)^2 - 2.7^2 \le 0 \\ 0 &\le x_1, x_2 \le 2 \end{aligned}$$

Example 2.

$$\begin{aligned} \min \quad f(x) &= x_1^2 + x_2^2 + 2x_3^2 + x_4^2 - 5x_1 - 5x2 - 21x_3 + 7x_4 \\ \text{s.t.} \quad g_1(x) &= 2x_1^2 + x_2^2 + x_3^2 + 2x_1 + x_2 + x_4 - 5 \le 0 \\ g_2(x) &= x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_1 - x_2 + x_3 - x_4 - 8 \le 0 \\ g3(x) &= x_1^2 + 2x_2^2 + x_3^2 + 2x_4^2 - x_1 - x_4 - 10 \le 0 \end{aligned}$$

Example 3.

$$\begin{aligned} \min \quad f(x) &= 1000 - x_1^2 - 2x_2^2 - x_3^2 - x_1x_2 - x_1x_3 \\ \text{s.t.} \quad g_1(x) &= x_1^2 + x_2^2 + x_3^2 - 25 = 0 \\ g_2(x) &= (x_1-5)^2 + x_2^2 + x_3^2 - 25 = 0 \\ g_3(x) &= (x_1-5)^2 + (x_2-5)^2 + (x_3-5)^2 - 25 \le 0 \\ 0 &\le x_i \le 100, i = 1, 2, 3. \end{aligned}$$

Example 4.

$$\begin{aligned} \min \quad f(x) &= 10x_2 + 2x_3 + x_4 + 3x_5 + 4x_6 \\ \text{s.t.} \quad g_1(x) &= x_1 + x_2 - 10 = 0 \\ g_2(x) &= -x_1 + x_3 + x_4 - x_5 = 0 \\ g_3(x) &= -x_2 - x_3 + x_5 + x_6 = 0 \\ g_4(x) &= 10x_1 - 2x_3 + 3x_4 - 2x_5 - 16 \le 0 \\ g_5(x) &= x_1 + 4x_3 + x_5 - 10 \le 0 \\ 0 &\le x_1 \le 12 \\ 0 &\le x_2 \le 18 \\ 0 &\le x_3 \le 5 \\ 0 &\le x_4 \le 12 \\ 0 &\le x_5 \le 1 \\ 0 &\le x_6 \le 16. \end{aligned}$$

References

[1] S. Wright, J. Nocedal, Numerical Optimization, vol. 2, Springer, New York, 1999.
[2] C. Wu, K.L. Teo, S. Wu, Min-max optimal control of linear systems with uncertainty and terminal state constraints, Automatica 49 (6) (2013) 1809–1815.
[3] K.F.C. Yiu, Y. Liu, K.L. Teo, A hybrid descent method for global optimization, J. Glob. Optim. 28 (2) (2004) 229–238.
[4] A. Levy, A. Montalvo, The tunneling algorithm for the global minimization of functions, SIAM J. Sci. Stat. Comput. 6 (1) (1985) 15–29.
[5] X. Liu, W. Xu, A new filled function applied to global optimization, Comput. Oper. Res. 31 (1) (2004) 61–80.
[6] C. Wu, K.L. Teo, V. Rehbock, H.H. Dam, Global optimum design of uniform FIR filter bank with magnitude constraints, IEEE Trans. Signal Process. 56 (11) (2008) 5478–5486.
[7] Q. Long, C. Wu, T. Huang, X. Wang, A genetic algorithm for unconstrained multi-objective optimization, Swarm Evol. Comput. 22 (2015) 1–14.
[8] Z. Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs, Springer, 1996.
[9] Q. Long, C. Wu, A hybrid method combining genetic algorithm and Hook-Jeeves method for constrained global optimization, J. Ind. Manag. Optim. 10 (4) (2014) 1279–1296.
[10] M.D. Toksari, Ant colony optimization for finding the global minimum, Appl. Math. Comput. 176 (1) (2006) 308–316.
[11] C. Wu, C. Li, Q. Long, A DC programming approach for sensor network localization with uncertainties in anchor positions, J. Ind. Manag. Optim. 10 (3) (2014) 817–826.
[12] S. Kirkpatrick, C.D. Gelatt, M.P. Vecchi, et al., Optimization by simulated annealing, Science 220 (4598) (1983) 671–680.
[13] F. Kang, J. Li, Z. Ma, Rosenbrock artificial bee colony algorithm for accurate global optimization of numerical functions, Inf. Sci. 181 (16) (2011) 3508–3531.
[14] H. Wang, J. Liu, Q. Wang, Modified Artificial Bee Colony algorithm for numerical function optimization, Comput. Eng. Appl. 48 (19) (2012) 36–39.
[15] J. Liu, H. Zhu, Q. Ma, Z. Lanlan, H. Xu, An Artificial Bee Colony algorithm with guide of global & local optima and asynchronous scaling factors for numerical optimization, Appl. Soft Comput. 37 (2015) 608–618.
[16] R.C. Eberhart, J. Kennedy, A new optimizer using particle swarm theory, in: Proceedings of the Sixth International Symposium on Micro Machine and Human Science, vol. 1, New York, NY, 1995, pp. 39–43.
[17] Y. Shi, R. Eberhart, A modified particle swarm optimizer, in: IEEE International Conference on Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence, IEEE, 1998, pp. 69–73.
[18] Z. Yan, J. Wang, G. Li, A collective neurodynamic optimization approach to bound-constrained nonconvex optimization, Neural Netw. 55 (2014) 20–29.
[19] X. Zhang, C. Wu, J. Li, X. Wang, Z. Yang, J.-M. Lee, K.-H. Jung, Binary artificial algae algorithm for multidimensional knapsack problems, Appl. Soft Comput. 43 (2016) 583–595.
[20] J. Liu, C. Wu, G. Wu, X. Wang, A novel differential search algorithm and applications for structure design, Appl. Math. Comput. 268 (2015) 246–269.
[21] J. Liu, K.L. Teo, X. Wang, C. Wu, An exact penalty function-based differential search algorithm for constrained global optimization, Soft Comput. (2015) 1–9.
[22] A. Mohammad Nezhad, R. Aliakbari Shandiz, A. Eshraghniaye Jahromi, A particle swarm-BFGS algorithm for nonlinear programming problems, Comput. Oper. Res. 40 (4) (2013) 963–972.
[23] V. Kelner, F. Capitanescu, O. Léonard, L. Wehenkel, A hybrid optimization technique coupling an evolutionary and a local search algorithm, J. Comput. Appl. Math. 215 (2) (2008) 448–456.
[24] B. Wu, C. Qian, W. Ni, S. Fan, Hybrid Harmony Search and Artificial Bee Colony algorithm for global optimization problems, Comput. Math. Appl. 64 (8) (2012) 2621–2634.
[25] W.T. Li, X.W. Shi, Y.Q. Hei, S.F. Liu, J. Zhu, A hybrid optimization algorithm and its application for conformal array pattern synthesis, IEEE Trans. Antennas Propag. 58 (10) (2010) 3401–3406.
[26] K. Miettinen, M.M. Mäkelä, H. Maaranen, Efficient hybrid methods for global continuous optimization based on simulated annealing, Comput. Oper. Res. 33 (4) (2006) 1102–1116.
[27] F. Kang, J. Li, H. Li, Artificial bee colony algorithm and pattern search hybridized for global optimization, Appl. Soft Comput. 13 (4) (2013) 1781–1791.
[28] S. Li, M. Tan, I.W. Tsang, J.-Y. Kwok, A hybrid PSO-BFGS strategy for global optimization of multimodal functions, IEEE Trans. Syst. Man Cybern. B: Cybern. 41 (4) (2011) 1003–1014.
[29] H.A. Bashir, R.S. Neville, Hybrid evolutionary computation for continuous optimization, 2013 arXiv:1303.3469.
[30] B.A. Tolson, C.A. Shoemaker, Dynamically dimensioned search algorithm for computationally efficient watershed model calibration, Water Resour. Res. 43 (1) (2007).
[31] Q. Lin, R. Loxton, K.L. Teo, Y.H. Wu, C. Yu, A new exact penalty method for semi-infinite programming problems, J. Comput. Appl. Math. 261 (2014) 271–286.
[32] C. Yu, K.L. Teo, L. Zhang, Y. Bai, A new exact penalty function method for continuous inequality constrained optimization problems, J. Ind. Manag. Optim. 6 (4) (2010) 895.
[33] C. Wu, K.L. Teo, Design of discrete Fourier transform modulated filter bank with sharp transition band, IET Signal Process. 5 (4) (2011) 433–440.
[34] D.C. Liu, J. Nocedal, On the limited memory BFGS method for large scale optimization, Math. Progr. 45 (1–3) (1989) 503–528.
[35] R.H. Byrd, J. Nocedal, R.B. Schnabel, Representations of quasi-Newton matrices and their use in limited memory methods, Math. Progr. 63 (1–3) (1994) 129–156.
[36] Y. Xiao, Z. Wei, Z. Wang, A limited memory BFGS-type method for large-scale unconstrained optimization, Comput. Math. Appl. 56 (4) (2008) 1001–1009.
[37] M.B. Reed, L-Broyden methods: a generalization of the L-BFGS method to the limited-memory Broyden family, Int. J. Comput. Math. 86 (4) (2009) 606–615.
[38] W. Zheng, P. Bo, Y. Liu, W. Wang, Fast B-spline curve fitting by L-BFGS, Comput. Aided Geom. Des. 29 (7) (2012) 448–462.
[39] M.S. Berkani, S. Giurgea, C. Espanet, J.-L. Coulomb, C. Kieffer, Study on optimal design based on direct coupling between a FEM simulation model and L-BFGS-B algorithm, IEEE Trans. Magn. 49 (5) (2013) 2149–2152.
[40] S. Ma, Y. Yang, H. Liu, A parameter free filled function for unconstrained global optimization, Appl. Math. Comput. 215 (10) (2010) 3610–3619.
[41] C. Gao, Y. Yang, B. Han, A new class of filled functions with one parameter for global optimization, Comput. Math. Appl. 62 (6) (2011) 2393–2403.
[42] S.-K.S. Fan, Y.-c. Liang, E. Zahara, Hybrid simplex search and Particle Swarm Optimization for the global optimization of multimodal functions, Eng. Optim. 36 (4) (2004) 401–418.
[43] S.-j. Lian, Smoothing approximation to l_1 exact penalty function for inequality constrained optimization, Appl. Math. Comput. 219 (6) (2012) 3113–3121.
[44] Z. Meng, Q. Hu, C. Dang, A penalty function algorithm with objective parameters for nonlinear mathematical programming, J. Ind. Manag. Optim. 5 (2009) 585–601.

[45] X. Xu, Z. Meng, J. Sun, R. Shen, A penalty function method based on smoothing lower order penalty function, J. Comput. Appl. Math. 235 (14) (2011) 4047–4058.
[46] M.Ç. Pinar, S.A. Zenios, On smoothing exact penalty functions for convex constrained optimization, SIAM J. Optim. 4 (3) (1994) 486–511.
[47] N. Baba, T. Shoman, Y. Sawaragi, A modified convergence theorem for a random optimization method, Inf. Sci. 13 (2) (1977) 159–166.
[48] D. Ortiz-Boyer, C. Hervás-Martí nez, N. Garcí a-Pedrajas, Cixl2: a crossover operator for evolutionary algorithms based on population features, J. Artif. Intell. Res. 24 (2005) 1–48.