Vilnius University
Institute of Data Science and
Digital Technologies
L I T H U A N I A

INFORMATICS (09 P)

IMPROVEMENT, DEVELOPMENT AND IMPLEMENTATION OF DERIVATIVE-FREE GLOBAL OPTIMIZATION ALGORITHMS

Linas Stripinis

October 2018

Technical Report MII-DS-09P-18-1 October 2016 - 30 September 2020

VU Institute of Mathematics and Informatics, Akademijos str. 4, Vilnius LT-08663, Lithuania
www.mii.lt

Abstract

We consider a box-constrained global optimization problem with a Lipschitz-continuous objective function and an unknown Lipschitz constant. The well known derivative-free global-search `DIRECT` (`DI`vide a hyper-`RECT`angle) algorithm performs well solving such problems. However, the efficiency of the `DIRECT` algorithm deteriorates on problems with many local optima and when the solution with high accuracy is required. To overcome these difficulties different regimes of global and local search are introduced or the algorithm is combined with local optimization. In first part we investigate a different direction of improvement of the `DIRECT` algorithm and propose a new strategy for the selection of potentially optimal rectangles, what does not require any additional parameters or local search subroutines. An extensive experimental investigation reveals the effectiveness of the proposed enhancements.

Applied optimization problems often include constraints. Although the well-known derivative-free global-search `DIRECT` algorithm performs well solving box-constrained global optimization problems, it does not naturally address constraints. We develop a new algorithm `DIRECT-GLce` for general constrained global optimization problems incorporating two-step selection procedure and penalty function approach in our recent `DIRECT-GL` algorithm. The proposed algorithm effectively explores hyper-rectangles with infeasible centers which are close to boundaries of feasibility and may cover feasible regions. An extensive experimental investigation revealed the potential of the proposed approach compared with other existing `DIRECT`-type algorithms for constrained global optimization problems, including important engineering problems.

Keywords: Global optimization, `DIRECT`-type algorithms, Derivative-free optimization, `DIRECT`-type constraint-handling, Nonconvex optimization

Contents

1 Introduction

We consider a box-constrained global optimization problem of the form

$$\min_{\mathbf{x}\in D} \quad f(\mathbf{x}) \tag{1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ denotes the objective function and the feasible region is an n-dimensional hyper-rectangle $D = [\mathbf{a}, \mathbf{b}] = \{\mathbf{x} \in \mathbb{R}^n : a_j \leq x_j \leq b_j, j = 1, \dots, n\}$. We also assume, that the objective function $f(\mathbf{x})$ is Lipschitz-continuous, but can be non-linear, non-differentiable, non-convex, and multi-modal. `DIRECT` is a popular partitioning-based Lipschitz optimization [HPT95,PŽ07,PŽ09,PŽ14,PŽG10,Pin96a,SS00] algorithm extending ideas of Piyavskii [Piy67] (independently rediscovered also by Shubert [Shu72]) algorithm to multidimensional derivative-free optimization. The `DIRECT` algorithm [JPS93] seeks a global optimum by partitioning potentially optimal (the most promising) hyper-rectangles and evaluating the objective function at the centers of these hyper-rectangles. Simplicity and efficiency of the `DIRECT` algorithm attracted considerable research interest. Although most of `DIRECT`-type algorithms use hyper-rectangular partitions [GK01,LC14,LZY15,LLP10a,LLP10b], simplicial partitions (`DISIMPL` algorithm) [PSKŽ14, PŽ13, PŽ14] have several advantages [PŽ16]. Central sampling of the objective function can be changed to diagonal approach sampling at the endpoints of diagonal [KPS03, SK06, SK08, SK17]. A trisection of hyper-rectangles is usually used to reuse the objective function values at the center or endpoints of diagonals in descendant subregions. However, a bisection can ensure better shapes of hyper-rectangles with a smaller variety of sizes in different dimensions than trisection which produces sizes differing by three times, but a specific sampling strategy is necessary to enable the reuse of sample points [PCŽ16].

2 Improved scheme for selection of potentially optimal hyper-rectangles in `DIRECT`

The original `DIRECT` algorithm has two main weaknesses [LYZZ17,LLP16,PSKŽ14,SK06]. First, on problems with many local minima, `DIRECT` sometimes spends an excessive number of function evaluations exploring suboptimal local minima, thereby delaying the discovery of the global minimum. To address this issue, a two-phase globally-biased technique was proposed [PSKŽ14, SK06]. Second, `DIRECT` usually gets close to the global optimum quickly, but it can be slow to converge with a high accuracy. To overcome the latter issue, a two-phase locally-biased technique [LZY15] or hybrid versions of `DIRECT`-type algorithms enriched with the use of local searches [LLP10a,LLP16] can be employed. In this section, we propose an alternative strategy to overcome both drawbacks without the need to use local solvers or use two-phase scheme which requires the introduction of new parameters.

2.1 The selection of the most promising hyper-rectangles

The essential step in `DIRECT`-type algorithms is identification of potentially optimal (the most promising) hyper-rectangles of the current partition, which at the iteration k is defined as

$$\mathcal{P}_k = \{D^i_k : i \in \mathbb{I}_k\},$$

where $D^i_k = [\mathbf{a}^i, \mathbf{b}^i] = \{\mathbf{x} \in \mathbb{R}^n : 0 \leq a^i_j \leq x_j \leq b^i_j \leq 1, j = 1, \dots, n, \forall i \in \mathbb{I}_k\}$ and $\mathbb{I}_k$ is the index set identifying the current partition $\mathcal{P}_k$. The next partition $\mathcal{P}_{k+1}$ is obtained after the subdivision of the selected potentially optimal hyper-rectangles from the current partition $\mathcal{P}_k$.

2.1.1 Potentially optimal hyper-rectangles in the original `DIRECT` algorithm

To make the selection of potentially optimal hyper-rectangles in the future iterations, `DIRECT` assesses the goodness based on the lower bound estimates for the objective function $f(\mathbf{x})$ over each hyper-rectangle D^i_k. The requirement of potential optimality is stated formally in Definition 1.

Definition 1 (Potentially optimal hyper-rectangle) *Let $\mathbf{c}^i$ denote the center sampling point and δ_i be a measure (distance, size) of the hyper-rectangle D^i_k. Let $\varepsilon > 0$ be a positive constant and $f_{\min}$ be the best currently known value of the objective function. A hyper-rectangle $D^j_k, j \in \mathbb{I}_k$ is said to be potentially optimal if there exists some rate-of-change (Lipschitz) constant $\tilde{L} > 0$ such that*

$$f(\mathbf{c}^j) - \tilde{L}\delta_j \leq f(\mathbf{c}^i) - \tilde{L}\delta_i, \quad \forall i \in \mathbb{I}_k, \tag{2}$$

$$f(\mathbf{c}^j) - \tilde{L}\delta_j \leq f_{\min} - \varepsilon|f_{\min}|, \tag{3}$$

where the measure of the hyper-rectangle is

$$\delta_i = \frac{1}{2}\|\mathbf{b}^i - \mathbf{a}^i\|_2. \tag{4}$$

The hyper-rectangle D^j_k is potentially optimal if the lower Lipschitz bound for the objective function computed by the left-hand side of (2) is the smallest one with some positive constant $\tilde{L}$ among the hyper-rectangles of the current partition $\mathcal{P}_k$. In (3) the parameter ε is used to protect from an excessive refinement of the local minima [JPS93, PSKŽ14].

2.1.2 Selection of the most promising hyper-rectangles in other `DIRECT`-type algorithms

In the original `DIRECT` algorithm, the size of a hyper-rectangle is measured by the Euclidean distance from its center to a corner or equivalently by a half length of a diagonal (see (4)). In `DIRECT-l` [GK01], the measure of a hyper-rectangle is instead evaluated by the length of its longest side. Such a measure corresponds to the L^∞-norm and allows the DIRECT-l algorithm to group more hyper-rectangles with the same measure. Thus, there are fewer distinct measures and therefore, less potentially optimal hyper-rectangles are selected. Moreover, in DIRECT-l at most one hyper-rectangle from each group is selected, even if there are more than one potentially optimal hyper-rectangle in the same group. This allows reduction of the number of divisions within a group. The results presented in [GK01] and extended in [PSKŽ14] suggest that DIRECT-l performs well for lower dimensional problems, which do not have too many local and global minima.

The main principle of an aggressive version of DIRECT [BWG+00] is to select and divide a hyper-rectangle of every measure (δ_i) in each iteration. The aggressive version requires many more function evaluations than the other versions of DIRECT since the criteria for choosing hyper-rectangles to be divided have been relaxed. Although this approach does not appear to be favorable for simple test problems, more difficult problems may be easier solved by this strategy on a large parallel supercomputer [BWG+00].

In the PLOR algorithm [MPR+17], the set of all Lipschitz constants (herewith the set of potentially optimal hyper-rectangles) is reduced to just two: the maximal and the minimal ones. In such a way the PLOR approach is independent of any user-defined parameters and balances equally local and global search during the optimization process.

A two-phase globally [PSKŽ14, SK06] and locally-biased [LZY15] algorithms at one of the phases work in the same as the original DIRECT algorithm, i.e., during the selection procedure considers all hyper-rectangles from the current partition. However, in the second phase, they limit the selection of potentially optimal hyper-rectangles based on their measures. The globally-biased versions constrain themselves to the larger subregions (primary addressing the first weakness), while the locally-biased version constrains itself to the smaller ones and in such a way addresses the second weakness of DIRECT-type algorithms.

2.2 Extended set of potentially optimal hyper-rectangles

In this section, we present a new way to identify the extended set of potentially optimal hyper-rectangles. Using a new two-step based strategy, we enlarge the set of the best hyper-rectangles by adding more medium-measured hyper-rectangles with the smallest function value at their centers and additionally, closest to the current minimum point. The first extension forces the algorithm to work more globally (compared to the selection procedure used in DIRECT), while the second part assures faster and broader examination around the current minimum point. In such way, we address both weaknesses of DIRECT staying in the same algorithmic framework. Let's state it formally.

Let $\mathbb{L}_k$ be the set of all different indices at the current partition $\mathcal{P}_k$, corresponding to the groups of hyper-rectangles having the same measure (δ_k). The minimum value $l_k^{\min} \in \mathbb{L}_k$ corresponds to the group of hyper-rectangles having the smallest measure $\delta_k^{\min}$. The maximum value $l_k^{\max}$ of $\mathbb{L}_k$ corresponds to the group of hyper-rectangles having the largest measures $\delta_k^{\max}$, i. e., $l_k^{\max} = \max\{\mathbb{L}_k\} < \infty$. Finally, let $l_k^i \in \mathbb{L}_k$ be the index of the group the hyper-rectangle D_k^i belongs to. Having this, in Definitions 2 and 3 we formalize new strategies for identification of an extended set of potentially optimal hyper-rectangles from the current partition $\mathcal{P}_k$.

Definition 2 (Enhancing the global search)

- ***Step 1*** *Find an index $j \in \mathbb{I}_k$ and a corresponding hyper-rectangle D_k^j, such that*

$$D_k^j = \arg\max_j \{l_k^j : j = \underset{i \in \mathbb{I}_k:\, l_k^{\min} \leq l_k^i \leq l_k^{\max}}{\arg\min} \{f(\mathbf{c}^i)\}\}. \tag{5}$$

- ***Step 2*** *Set $l_k^{\min} = l_k^j + 1$. If $l_k^j \leq l_k^{\max}$* ***repeat*** *from Step 1; otherwise* ***terminate****.*

At Step 1, the hyper-rectangle containing the minimum point $(\mathbf{x}^{\min})$ is selected. If there are several hyper-rectangles with the same lowest objective value $f(\mathbf{c}^i)$, the preference is given to hyper-rectangles with the largest l_k^j value, i.e., a bigger size measure. After this, in Step 2, the minimum value $l_k^{\min} = l_k^j + 1$ is increased; thus all hyper-rectangles from the groups with indices lower than the updated $l_k^{\min}$ (measures of these hyper-rectangles belonging to these groups are smaller than the measure of the $l_k^{\min}$ group) are not considered in the recurrent Step 1. A geometrical interpretation and comparison of the original `DIRECT` and the globally enhanced (let us call `DIRECT-G`) versions are shown in the left-hand side and middle graphs in Figure 1. By this strategy, we extend the number of medium-measured potentially optimal hyper-rectangles and force `DIRECT-G` to work more globally. Let us stress, that opposed to the aggressive `DIRECT` version, by Definition 2 `DIRECT-G` will not consider hyper-rectangles from the groups where the minimum function value is larger compared to the minimum value from the larger groups.

Definition 3 (Enhancing the local search)

- ***Step 1*** *At each iteration k, evaluate the Euclidean distance from the current minimum point $(\mathbf{x}^{\min})$ to other sampled points:*

$$d(\mathbf{x}^{\min}, \mathbf{c}^i) = \sqrt{\sum_{j=1}^{n} (x_j^{\min} - c_j^i)^2} \tag{6}$$

- ***Step 2*** *Apply the procedure described in Definition 2 in (5) using distances $d(\mathbf{x}^{\min}, \mathbf{c}^i)$ instead of objective function values.*

A geometrical interpretation of the selection of potentially optimal hyper-rectangles using the locally enhanced strategy is shown on the right-hand side of Figure 1. By this strategy, we extend the number of potentially optimal hyper-rectangles locating close to the current minimum point $(\mathbf{x}^{\min})$. Moreover, by this strategy, we select the closest hyper-rectangles from various measures.

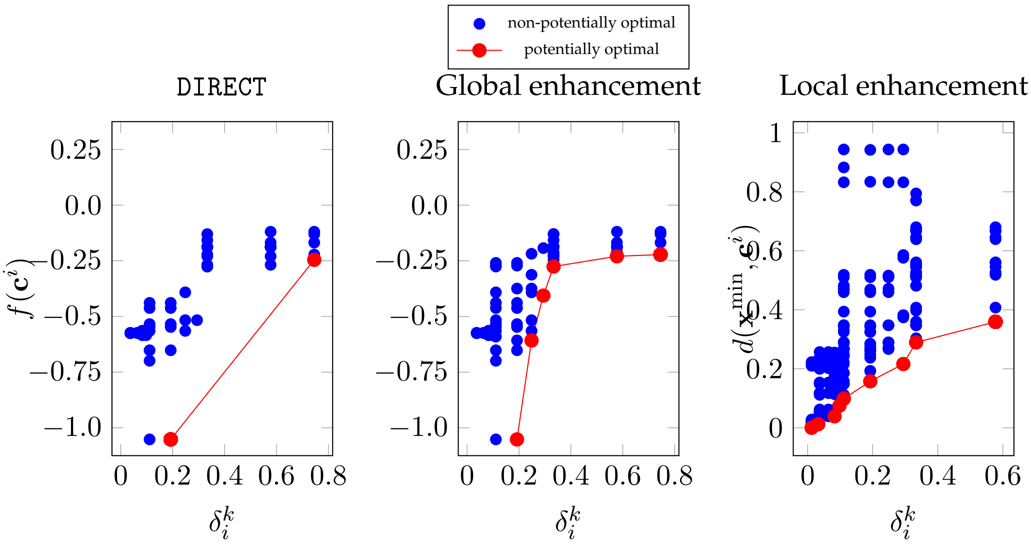

Figure 1: Geometric interpretation of the selection of potentially optimal hyper-rectangles by using `DIRECT` (on the left-hand side), `DIRECT-G` (middle), and the locally enhanced strategy (on the right-hand side) on the Shekel 5 test problem in the fifth iteration of corresponding algorithms/strategies

2.2.1 `DIRECT-GL` algorithm

In this subsection, we introduce a new `DIRECT`-type algorithm (let us call `DIRECT-GL`). The key feature of `DIRECT-GL` is that `DIRECT-GL` performs the identification of potentially-optimal hyper-rectangles twice in every iteration. First, by using Definition 2 the globally enhanced set of potentially optimal candidates is determined and fully processed (sampled and partitioned). Second, by using Definition 3 the locally enhanced set is identified and fully processed (sampled and partitioned) again. Thus, our new approach is based on "Divide the best" strategy [Ser98] and it has the everywhere-dense type of convergence (like other `DIRECT`-type algorithms [FK06, JPS93, PCŽ16, PSKŽ14, SK06]). This follows from the fact that, that using Definitions 2 and 3, `DIRECT-GL` always selects for partitioning hyper-rectangles from the group ($l_k^{\max}$) with the largest measure $\delta_k^{\max}$. Since each group contains only a finite number of hyper-rectangles, after a sufficient number of iterations, all hyper-rectangles will be partitioned. Such a procedure will be repeated with a new group of the largest hyper-rectangles and so on until the largest hyper-rectangles will have the measure smaller than the required tolerance ε.

The complete description of the `DIRECT-GL` algorithm is shown in Algorithm 1. The input for the algorithm is one (or few) stopping criteria: required tolerance (ε_{pe}), the maximal number of function evaluations ($\mathtt{M}_{\max}$) and the maximal number of `DIRECT-GL` iterations ($\mathtt{K}_{\max}$). After termination, `DIRECT-GL` returns the found objective value $f_{\min}$ and the solution point $\mathbf{x}^{\min}$ together with algorithmic performance measures: final tolerance – percent error (pe), the number of function evaluations (m), and the number of iterations (k).

input : ε_{pe}, $\mathtt{M}_{\max}$, $\mathtt{K}_{\max}$;
output: $f_{\min}$, $\mathbf{x}^{\min}$;

1 Initialize $k = 1$, $m = 1$, $\mathbb{I}_{\text{k}} = \{1\}$, $f_{\min} = f(\mathbf{c}^1)$, $\mathbf{x}^{\min} = \mathbf{c}^1$;
2 **while** $pe > \varepsilon_{\text{pe}}$ **and** $m < \mathsf{M}_{\max}$ **and** $k < \mathsf{K}_{\max}$ **do** `// pe defined in Eq.` (7)
3   Identify the index set $\mathbb{J}_k^1 \subseteq \mathbb{I}_k$ of potentially optimal hyper-rectangles using Definition 2;
4   Set $\mathbf{x}_{\text{old}}^{\min} = \mathbf{x}^{\min}$;
5   **foreach** $i \in \mathbb{J}_k^1$ **do**
6   Subdivide (trisect) hyper-rectangle D_k^i and update $\mathbb{I}_{\text{k}}$;
7   Evaluate f at the centers of the new hyper-rectangles;
8   Update $f_{\min}$, $\mathbf{x}^{\min}$, pe and m;
9   **end**
10   **if** $\mathbf{x}^{\min} \neq \mathbf{x}_{\text{old}}^{\min}$ **then**
11   Calculate distances $\mathbf{d}(\mathbf{x}^{\min}, \mathbf{c}^{\text{i}})$, $i \in \mathbb{I}_{\text{k}}$ to all sampled points; `// using Eq.` (6)
12   Set $\mathbf{x}_{\text{old}}^{\min} = \mathbf{x}^{\min}$;
13   **else**
14   Calculate distances $\mathbf{d}(\mathbf{x}^{\min}, \mathbf{c}^{\text{i}})$ to newly sampled points;
15   **end**
16   Identify the index set $\mathbb{J}_{\text{k}}^2 \subseteq \mathbb{I}_k$ of potentially optimal hyper-rectangles using Definition 3;
17   **foreach** $i \in \mathbb{J}_k^2$ **do**
18   Subdivide (trisect) hyper-rectangle D_k^i and update $\mathbb{I}_{\text{k}}$;
19   Evaluate f at the centers of the new hyper-rectangles;
20   Update $f_{\min}$, $\mathbf{x}^{\min}$, pe and m;
21   **end**
22   Increase $k = k + 1$ and check **if** condition described in lines 10-15;
23 **end**
24 **return** $f_{\min}$, $\mathbf{x}^{\min}$, pe, k, m;

Algorithm 1: Pseudo code of the `DIRECT-GL` algorithm

2.2.2 Numerical investigation

The introduced DIRECT-G and DIRECT-GL as well as the original DIRECT algorithm (Finkel's implementation [Fin04]) were implemented in the MATLAB programming language. Note, that for the DIRECT algorithm potentially optimal hyper-rectangles can be identified in at least two different ways: using modified Graham's scan algorithm [BH99] or the rule described by Lemma 2.3 in [Gab01]. Usually this does not impose significant differences, but occasionally it can have, e.g., when a higher precision is required. The selection procedure of potentially optimal hyper-rectangles in DIRECT-GL differs significantly, however, this does not have a notable difference to the overall performance, compared with the procedure used in DIRECT. This means, that for the identification of the same quantity of potentially optimal hyper-rectangles DIRECT and DIRECT-GL spent a similar amount of time.

We compare the efficiency of the algorithms on the Hedar test set [Hed05], which consist of 27 global optimization test functions. Some of test problems have several variants, e.g., Bohachevsky, Hartman, Shekel, and some of them can be tested for different dimensionality. In Table 1 we report main features of these problems: problem number (No.), name, dimensionality (n), feasible region (D), the number of local minima (if known), and the known minimum (f^*). Whenever the global minimum point lies at the initial sampling point for any tested algorithm the feasible region was modified (increased). These modified problems are marked with the star sign *.

Note, that the most of test problems from the Hedar test set are multimodal, therefore suitable to investigate how introduced modifications help to overcome the first weakness. Since all the global minima f^* are known for all Hedar test problems in advance, investigated algorithms were stopped either when the point $\bar{\mathbf{x}}$ was generated such that the percent error

$$pe = 100\% \times \begin{cases} \frac{f(\bar{\mathbf{x}})-f^*}{|f^*|}, & f^* \neq 0, \\ f(\bar{\mathbf{x}}), & f^* = 0, \end{cases} \tag{7}$$

is smaller than the tolerance value ε_{pe}, or when the number of function evaluations exceeds the prescribed limit of 10^6. In our investigation, four different values for ε_{pe} were considered: 10^{-2}, 10^{-4}, 10^{-6}, 10^{-8}. By doing this, we investigate algorithm's ability to avoid the second weakness. The comparison is based on the number of function evaluations and the best (smallest) number for each problem is shown in bold font.

The results of experiments are given in Table 2. First, observe that DIRECT-G and DIRECT-GL perform on average much better (see **Aver. (overall)**) compared to DIRECT. Especially this is evident when a lower percentage error (pe) (higher accuracy) is sought. Observe, that original DIRECT on average performs better only for simpler (unimodal) test problems (see **Aver. (unimodal)**). That is mainly because the set of potentially optimal hyper-rectangles in DIRECT-G, DIRECT-L and DIRECT-GL is larger per iteration. Consequently, a greater number of function evaluations is needed.

Table 1: Key characteristics of the Hedar test problems

Problem No.	Problem name	Dimension n	Feasible region D	No. of local minima	Optimum f^*
$1, 2, 3$	Ackley*	2, 5, 10	$[-15, 35]^n$	multimodal	0.0
4	Beale	2	$[-4.5, 4.5]^2$	multimodal	0.0
5	Bohachevsky 1*	2	$[-100, 110]^2$	multimodal	0.0
6	Bohachevsky 2*	2	$[-100, 110]^2$	multimodal	0.0
7	Bohachevsky 3*	2	$[-100, 110]^2$	multimodal	0.0
8	Booth	2	$[-10, 10]^2$	unimodal	0.0
9	Branin	2	$[-5, 10] \times [10, 15]$	3	0.39789
10	Colville	4	$[-10, 10]^4$	multimodal	0.0
$11, 12, 13$	Dixon & Price	2, 5, 10	$[-10, 10]^n$	unimodal	0.0
14	Easom	2	$[-100, 100]^2$	multimodal	-1.0
15	Goldstein & Price	2	$[-2, 2]^2$	4	3.0
16	Griewank*	2	$[-600, 700]^2$	multimodal	0.0
17	Hartman	3	$[0, 1]^3$	4	-3.86278
18	Hartman	6	$[0, 1]^6$	4	-3.32237
19	Hump	2	$[-5, 5]^2$	6	-1.03163
$20, 21, 22$	Levy	2, 5, 10	$[-10, 10]^n$	multimodal	0.0
23	Matyas*	2	$[-10, 15]^2$	unimodal	0.0
24	Michalewicz	2	$[0, \pi]^2$	2!	-1.80130
25	Michalewicz	5	$[0, \pi]^5$	5!	-4.68765
26	Michalewicz	10	$[0, \pi]^{10}$	10!	-9.66015
27	Perm	4	$[-4, 4]^4$	multimodal	0.0
$28, 29$	Powell	4, 8	$[-4, 5]^n$	multimodal	0.0
30	Power Sum	4	$[0, 4]^4$	multimodal	0.0
$31, 32, 33$	Rastrigin*	2, 5, 10	$[-5.12, 6.12]^n$	multimodal	0.0
$34, 35, 36$	Rosenbrock	2, 5, 10	$[-5, 10]^n$	unimodal	0.0
$37, 38, 39$	Schwefel	2, 5, 10	$[-500, 500]^n$	unimodal	0.0
40	Shekel, $m = 5$	4	$[0, 10]^4$	5	-10.15320
41	Shekel, $m = 7$	4	$[0, 10]^4$	7	-10.40294
42	Shekel, $m = 10$	4	$[0, 10]^4$	10	-10.53641
43	Shubert	2	$[-10, 10]^2$	760	-186.73091
$44, 45, 46$	Sphere*	2, 5, 10	$[-5.12, 6.12]^n$	multimodal	0.0
$47, 48, 49$	Sum squares*	2, 5, 10	$[-10, 15]^n$	unimodal	0.0
50	Trid	6	$[-36, 36]^6$	multimodal	-50.0
51	Trid	10	$[-100, 100]^{10}$	multimodal	-210.0
$52, 53, 54$	Zakharov*	2, 5, 10	$[-5, 11]^n$	multimodal	0.0

For small dimensional problems (see **Aver.** $(n \leq 3)$), DIRECT requires on average from 4.5 times (when $\varepsilon_{\text{pe}} = 10^{-2}$) to 175 times more function evaluations (when $\varepsilon_{\text{pe}} = 10^{-8}$) compared to DIRECT-GL. Also DIRECT-L showed an advantage comparing with DIRECT. Observe, that DIRECT-G performed worst with $\varepsilon_{\text{pe}} = 10^{-2}$ and $\varepsilon_{\text{pe}} = 10^{-4}$. Again, for most of these problems DIRECT was able to converge after a small number of iterations. Therefore, by extending the set of potentially optimal hyper-rectangles only globally enhanced (DIRECT-G) is not very efficient for low-dimensional problems. However, when $\varepsilon_{\text{pe}} = 10^{-6}$ and $\varepsilon_{\text{pe}} = 10^{-8}$ was used, DIRECT-G performed significantly better compared to DIRECT.

For higher dimensional (see **Aver.** $(n \geq 4)$) and multimodal problems (see **Aver. (multimodal)**) both introduced versions performed significantly better compared to DIRECT, and the best results were obtained using DIRECT-GL. Finally, in total DIRECT failed for 30.1% (65/216) cases, most of which when a lower percent error tolerance was required (10^{-6} and 10^{-8}) and optimization problems were more challenging. Meanwhile, DIRECT-G, DIRECT-L and DIRECT-GL in total failed on 18.1% (39/216),24% (52/216) and 9.2% (20/216) cases, accordingly.

Table 2: Number of function evaluations using `DIRECT`, `DIRECT-G` and `DIRECT-GL` algorithms solving Hedar test problems

Problem	`DIRECT`				`DIRECT-G`				`DIRECT-L`				`DIRECT-GL`			
No./$\varepsilon_{\rm pe}$	10^{-2}	10^{-4}	10^{-6}	10^{-8}	10^{-2}	10^{-4}	10^{-6}	10^{-8}	10^{-2}	10^{-4}	10^{-6}	10^{-8}	10^{-2}	10^{-4}	10^{-6}	10^{-8}
1	**225**	**443**	**655**	**909**	773	1,385	2,301	3,463	751	1,343	2,239	3,377	1,197	2,123	3,571	5,415
2	**8,845**	**11,289**	**14,619**	**17,757**	10,611	19,137	31,459	47,065	138,165	146,359	158,897	174,231	19,403	35,175	55,843	84,979
3	**80,927**	$> 10^6$	$> 10^6$	$> 10^6$	90,089	**151,575**	**240,677**	**350,075**	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	180,707	306,089	486,459	702,121
4	655	1,143	1,823	2,835	283	591	891	1,347	357	721	1,119	1,615	**183**	**395**	**591**	**833**
5	**327**	**457**	**551**	**845**	435	607	739	1,129	435	611	743	1,133	729	847	1,115	1,767
6	**345**	**489**	**589**	**897**	441	617	749	1,139	855	1,025	1,155	1,545	727	845	1,113	1,765
7	693	1,073	1,645	2,099	623	935	1,407	2,057	**459**	**787**	**1,119**	**1,595**	685	1,113	1,665	2,139
8	295	511	917	1,295	301	489	901	1,221	**283**	**395**	**699**	**1,015**	345	509	831	1,087
9	**195**	377	38,455	$> 10^6$	255	**365**	**603**	**841**	333	457	755	1,079	333	579	859	1,239
10	6,585	18,261	24,485	67,695	104,315	120,077	128,847	162,751	9,465	18,915	21,405	23,197	**1,623**	**2,809**	**3,539**	**5,371**
11	481	597	1,143	1,969	403	477	973	1,489	373	537	971	1,349	**235**	**393**	**823**	**1,297**
12	18,237	19,407	23,065	32,229	14,531	17,135	23,955	**29,471**	213,759	215,109	221,133	230,409	**13,109**	**16,501**	**22,951**	31,213
13	**365,221**	**458,743**	$> 10^6$	$> 10^6$	990,493	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$
14	32,859	59,347	297,571	$> 10^6$	336,879	337,069	337,169	337,477	**377**	**623**	**741**	**1,097**	495	817	1,085	1,679
15	**191**	**305**	10,437	$> 10^6$	209	357	**553**	789	269	415	603	839	223	367	555	**789**
16	9,215	9,341	9,341	9,505	12,519	12,711	12,711	12,965	**1,753**	**1,965**	**1,965**	**2,249**	2,067	2,375	2,375	2,799
17	**199**	4,165	88,883	$> 10^6$	369	**669**	819	**1,493**	325	**621**	931	1,623	379	1,049	1,199	2,431
18	**571**	182,623	$> 10^6$	$> 10^6$	1,529	**4,063**	**6,903**	**12,163**	1,557	4,249	7,027	12,237	4,793	8,793	13,207	19,879
19	293	997	54,487	$> 10^6$	**211**	**355**	593	965	**211**	359	**555**	**927**	279	485	657	1,143
20	**127**	**155**	**267**	**401**	189	225	407	585	149	221	399	577	189	263	459	581
21	**705**	**1,021**	**1,921**	**2,845**	1,587	2,563	4,325	6,253	1,533	2,485	4,193	6,101	2,349	4,361	6,329	10,149
22	**5,589**	**10,431**	**18,475**	**28,461**	11,149	18,801	30,673	44,013	10,303	17,555	28,761	41,505	16,179	29,945	48,049	74,815
23	107	209	391	935	111	225	379	825	**65**	**179**	**281**	**477**	101	211	357	557
24	**67**	**109**	**109**	**109**	97	179	179	179	97	179	179	179	129	235	235	235
25	14,077	215,127	$> 10^6$	$> 10^6$	5,491	7,105	7,819	7,819	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	**2,445**	**4,619**	**5,575**	**5,575**
26	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	**601,433**	**608,113**	**611,077**	**611,077**
27	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$
28	13,675	67,515	309,427	$> 10^6$	11,589	50,149	320,073	$> 10^6$	**5,135**	34,179	321,343	$> 10^6$	7,045	**24,591**	**85,235**	**202,795**
29	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	364,693	$> 10^6$	$> 10^6$	$> 10^6$	**147,105**	**905,027**	$> 10^6$	$> 10^6$
30	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	**13,243**	$> 10^6$	$> 10^6$	$> 10^6$	101,181	**763,635**	$> 10^6$	$> 10^6$
31	987	1,181	1,565	1,833	2,897	3,087	3,333	3,631	24,883	25,053	25,327	25,533	**811**	**1,109**	**1,507**	**1,803**
32	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	**180,429**	**184,247**	**192,151**	**196,343**
33	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$

Continued on next page

Table 2 Continued from previous page

Problem	DIRECT				DIRECT-G				DIRECT-L				DIRECT-GL			
No./ε_{pe}	10^{-2}	10^{-4}	10^{-6}	10^{-8}	10^{-2}	10^{-4}	10^{-6}	10^{-8}	10^{-2}	10^{-4}	10^{-6}	10^{-8}	10^{-2}	10^{-4}	10^{-6}	10^{-8}
34	1,621	1,913	3,005	4,019	389	619	2,285	3,883	**313**	**471**	**679**	**1,471**	579	727	1,143	1,657
35	**19,693**	**24,681**	**35,575**	**41,687**	20,363	28,293	46,005	68,065	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	25,395	38,633	72,735	86,043
36	169,191	215,435	267,741	308,715	**53,193**	**83,559**	**146,087**	**273,021**	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	95,405	167,319	268,591	403,207
37	**255**	**447**	**597**	**1,195**	371	567	691	1,153	807	989	1,105	1,555	659	971	1,235	1,709
38	**27,543**	**30,307**	**31,199**	39,487	637,379	640,081	640,743	645,519	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	556,495	561,599	562,903	568,483
39	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$
40	**155**	**255**	$> 10^6$	$> 10^6$	781	1,419	2,477	3,803	731	1,365	**2,389**	**3,697**	1,227	2,025	3,433	5,209
41	**145**	4,875	$> 10^6$	$> 10^6$	755	2,017	3,737	5,377	697	**1,953**	**3,645**	**5,273**	1,141	2,845	4,741	6,623
42	**145**	4,939	$> 10^6$	$> 10^6$	715	1,977	3,493	5,111	709	**1,949**	**3,443**	**5,047**	1,151	2,871	4,789	7,137
43	2,967	3,867	68,667	$> 10^6$	4,089	4,219	4,393	4,603	**369**	**535**	**807**	**1,079**	425	735	951	1,341
44	209	417	633	1,211	191	337	481	785	**173**	**309**	**449**	**743**	391	549	737	1,103
45	4,653	10,583	20,123	44,099	**2,287**	4,113	6,335	10,933	2,573	**3,963**	**6,103**	**10,175**	4,357	8,249	11,011	18,225
46	99,123	205,013	614,749	$> 10^6$	**16,857**	**28,243**	47,529	76,723	20,115	28,727	**46,803**	**75,211**	35,721	63,399	94,991	155,511
47	**107**	**195**	**321**	**623**	143	251	391	705	143	251	391	567	191	337	525	759
48	**833**	**1,489**	**2,463**	**3,827**	1,951	3,271	5,267	7,745	1,857	3,165	5,153	7,237	2,919	4,701	7,523	11,031
49	**7,795**	**14,691**	**22,651**	**34,735**	16,523	24,489	37,645	53,647	13,563	22,427	34,919	48,637	24,763	41,781	63,413	89,543
50	**4,897**	207,399	$> 10^6$	$> 10^6$	5,077	**10,069**	**17,411**	**26,079**	12,149	23,015	42,051	60,457	7,795	15,735	26,059	38,929
51	66,615	$> 10^6$	$> 10^6$	$> 10^6$	**22,201**	251,255	$> 10^6$	$> 10^6$	261,301	608,797	742,935	$> 10^6$	36,525	**119,093**	**174,059**	**299,163**
52	**237**	**303**	**653**	**949**	295	329	709	1,023	249	281	605	779	345	413	889	1,123
53	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	377,737	602,319	613,251	$> 10^6$	**5,465**	**9,725**	**15,591**	**22,243**	6,429	9,967	17,665	23,891
54	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	**94,175**	**151,287**	**268,999**	**317,611**	115,073	184,033	320,267	394,467
Aver. (overall)	184,591	236,891	369,800	493,577	199,253	211,822	235,896	263,322	226,023	265,436	277,382	298,068	**114,887**	**150,622**	**170,131**	**186,799**
Aver. (unimodal)	**115,099**	**126,330**	**170,648**	**176,480**	195,439	199,961	207,523	220,482	373,655	374,537	376,095	378,051	194,300	202,406	214,502	228,328
Aver. (multimodal)	208,913	275,588	439,503	604,561	200,588	215,973	245,826	278,316	174,351	227,251	242,832	270,074	**87,092**	**132,498**	**154,601**	**172,263**
Aver. ($n \leq 3$)	2,290	3,828	25,335	262,245	15,760	15,942	16,246	16,685	1,480	1,666	1,905	2,278	**509**	**759**	**1,064**	**1,533**
Aver. ($n \geq 4$)	319,846	409,809	625,371	665,211	335,394	357,192	398,862	446,311	392,619	461,136	481,767	517,525	**199,748**	**261,812**	**295,568**	**324,254**
Failed	9	11	18	26	8	9	10	12	11	13	13	15	**4**	**4**	**6**	**6**

Concluded

3 Penalty functions and two-step selection procedure based `DIRECT`-type algorithm for constrained global optimization

Many constrained optimization problems are formed from an engineering design process, where systems are often modeled with nonlinear and multimodal behavior, being low or high dimensional, computationally cheap or expensive. Another difficulty of real world engineering problems is constraints, which often allow feasible solutions only in a small subset of the design space, or split the feasible region in many non-intersecting subsets. In most cases practical engineering problems are complex and difficult to solve by traditional optimization methods. Many real world problems in engineering and applied sciences can be formulated as nonlinear programming global optimization problems [BG04, Flo99, Pin96b, SW10b].

We are seeking the global solution of the general nonlinear programming problem:

$$\begin{aligned} &\min_{\mathbf{x} \in D} f(\mathbf{x}) \\ &\text{s.t. } \mathbf{g}(\mathbf{x}) \leq \mathbf{0}, \\ &\quad\ \ \mathbf{h}(\mathbf{x}) = \mathbf{0}, \end{aligned} \tag{8}$$

where $f : \mathbb{R}^n \to \mathbb{R}, \mathbf{g} : \mathbb{R}^n \to \mathbb{R}^m, \mathbf{h} : \mathbb{R}^n \to \mathbb{R}^r$ are (possibly nonlinear) continuous functions and $D = [\mathbf{a}, \mathbf{b}] = \{\mathbf{x} \in \mathbb{R}^n : a_j \leq x_j \leq b_j, j = 1, \dots, n\}$.

The feasible region consisting of points that satisfy all the constraints is denoted by $D^{\text{feas}} = D \cap \Omega$, where $\Omega = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{g}(\mathbf{x}) \leq \mathbf{0} \text{ and } \mathbf{h}(\mathbf{x}) = \mathbf{0}\}$. We also assume, that all functions are Lipschitz-continuous (with unknown Lipschitz constants), but can be non-linear, non-differentiable, and non-convex. The original `DIRECT` algorithm [JPS93], as well as various modifications [LC14, PCŽ16, PSKŽ14, PŽ13, PŽ14, SK06], addresses optimization problems only with bounds on the variables. The first `DIRECT`-type approach for problems with general constraints was proposed by one of the original `DIRECT` authors [Jon01]. A few years later, the comparison of three different constraint handling strategies withing the `DIRECT` framework was carried out [Fin05]. The first three strategies revealed disadvantages of handling infeasible hyper-rectangles and opened many ways for researchers to improve existing and create new strategies. Only in recent years, several promising extensions of the original `DIRECT` algorithm have been proposed [BDLM12, CRF17, LXC+17, PLR10, PLL+16] for general engineering global optimization problems.

In this paper we introduce the extension for general engineering optimization problems to our recently proposed `DIRECT-GL` [SPŽ17] algorithm, which is based on a new strategy (compared to the most of `DIRECT`-type methods) for the selection of the extended set of potentially optimal hyper-rectangles (POH). The proposed `DIRECT-GLce` algorithm uses an auxiliary function approach, that combines information on the objective and constraint functions and does not require any penalty parameters. The `DIRECT-GLce` algo-

Table 3: Summary of the main algorithmic characteristics of DIRECT-type methods for (8) optimization problem

Step/Algorithm	DIRECT-L1	eDIRECT-C	filter-based DIRECT	DIRECT-GLce
Selection of potentially optimal hyper-rectangles (POH)	Original DIRECT strategy	Novel DIRECT-type constraint-handling technique that separately handles feasible and infeasible cells	Modified strategy, uses three sets: one from feasible, one from infeasible non-dominated and one from infeasible dominated points	Uses two step selection procedure from DIRECT-GL algorithm [SPŽ17]
Partitioning scheme	Original DIRECT trisection strategy	Based on Voronoi diagrams for partition the design space in Voronoi cells	Trisection strategy using the rules of "preference point" and "preference order" described in Definition 5 [CRF17]	Original DIRECT trisection strategy
Local minimization procedure	–	In MATLAB implementation uses fmincon	–	Only in the version: DIRECT-GLce-min
Input parameters	Balance parameter ϵ, penalty parameters p_i	Balance parameter ϵ, acceptable constraint violation ε_φ	Balance parameter ϵ, filter control parameters, acceptable constraint violation ε_φ	Acceptable constraint violation ε_φ

rithm works in two phases, where during the first phase the algorithm handles infeasible initial points while in the second phase seeks to find a feasible global solution. A separate phase for handling infeasible initial points is especially useful when the feasible region is small compared to the design space. When feasible solutions are located the efforts may be switched to finding better feasible solutions.

3.1 DIRECT-type methods for general optimization problem

In this subsection, we review and summarize existing DIRECT-type methods for (8) optimization problems.

The first DIRECT-type approach for problems with general constraints was presented in [Jon01]. The author extended the original DIRECT algorithm to handle nonlinear inequality constraints by using an auxiliary function that combines information on the objective and constraint functions in a special manner.

Second DIRECT-type approach is based the Neighborhood Assignment Strategy (NAS) [Gab01]. The idea of this strategy is to change the value of the objective function at the infeasible point $\bar{\mathbf{x}} \notin D^{\text{feas}}$ with the objective value attained in the feasible point from the neighborhood of $\bar{\mathbf{x}}$. Such a strategy does not allow the DIRECT algorithm to move beyond the feasible region. As the NAS strategy does not use all the available information, such as constraint violations, it is slower in general compared to other approaches and should be used only for optimization problems with hidden constraints.

Another strategy is based on the exact L1 penalty functions [Fle87]. An exact L1 penalty approach is a transformation of the original constrained problem (8) to the form:

$$\min_{\mathbf{x} \in D} f(\mathbf{x}) + \sum_{i=1}^{m} \max\{p_i g_i(\mathbf{x}), 0\} + \sum_{i=1}^{r} p_{i+m} |h_i(\mathbf{x})|, \tag{9}$$

where p_i are penalty parameters. Comparison in [Fin05] showed promising results. The biggest drawback is the requirement for the users to set penalty parameters for each constraint function. In practice, choosing penalty parameters is very important task and can have a huge impact on the performance of the algorithm [Fin05, LXC+17, PŽ14, PŽ16].

Recently, two new approaches based on penalty functions were proposed: `EPGO` [PLR10] and `DF-EPGO` [PLL+16]. The main feature of these algorithms is an automatic update rule for the penalty parameter and under the weak assumptions, the penalty parameters are updated only a finite number of times. Another recently proposed `DIRECT`-type approach filter-based `DIRECT` [CRF17] aims to minimize the constraint violations and the objective function value simultaneously. While other strategies work only with one general set of all hyper-rectangles, filter-based `DIRECT` algorithm adapts filter methodology from [FL02] and splits the main set into three separate sets. The filter strategy prioritizes the selection of potentially optimal candidates: first hyper-rectangles with feasible center points are selected, followed by those with infeasible but non-dominated center points, and finally by those that have infeasible and dominated center points.

A metamodel-based [FK09,SW10a,SW10b] constrained `DIRECT`-type global optimization algorithm (`eDIRECT-C`) was recently also proposed in [LXC+17]. One of the main differences and features of the algorithm is employed Voronoi diagrams for partitioning the design space in Voronoi cells. Voronoi cells have irregular boundaries and `eDIRECT-C` generates a set of random points to describe the cells. In order to speed up the convergence, the algorithm employs a pure greedy search on the objective metamodel $\hat{f}$. Also `eDIRECT-C` separately handles feasible and infeasible cells.

The summary of discussed and proposed algorithms is presented in Table 3.

3.2 Experimental investigation of the exact L1 penalty strategy within `DIRECT-GL` algorithm

In [SPŽ17], the comparison of `DIRECT-GL` algorithm against the original `DIRECT` as well as several other well-known `DIRECT`-type approaches was carried out on a class of well-known box-constrained global optimization test problems from [Hed05]. The results revealed, that for simpler (lower dimensional and unimodal) problems the original `DIRECT` algorithm performs well, but for more challenging (higher dimensional and multimodal) problems `DIRECT-GL` performs significantly faster compared to other tested `DIRECT`-type approaches. Motivated by the potential of the `DIRECT-GL` algorithm, we integrate the exact L1 penalty function strategy within `DIRECT-GL` and call the extended algorithm `DIRECT-GL-L1`. In the first implementation, for each constraint the penalty parameters for L1 functions are kept fixed during the optimization process. Analogously to [PŽ14] we use three different penalty parameters ($p = 10$, $p = 10^2$, and $p = 10^3$) for all constraint functions. Algorithmic comparison was carried out using a collection of 56 generally constrained test problems. Key characteristics of the used optimization test problems are summarized in Appendix Nr. 1., Table 13. Description of all test problems used in this and subsequent section in a Matlab format is provided in the online resource [SP18]. Note that problem G12* has the global minimum point in the center of the feasible region, thus we have modified bound constraints in the same way as in [LXC+17]. Since all the global minima f^* are known for all collected test problems in advance, tested algorithms were stopped either when a point $\bar{\mathbf{x}}$ was generated such that the percent error

$$pe \leq \varepsilon_{\mathrm{pe}}, \tag{10}$$

where

$$pe = \begin{cases} \frac{f(\bar{\mathbf{x}}) - f^*}{|f^*|}, & f^* \neq 0, \\ f(\bar{\mathbf{x}}), & f^* = 0, \end{cases}$$

often $\varepsilon_{\mathrm{pe}} = 10^{-4}$, or when the number of function evaluations exceeds the prescribed limit of 10^6.

Table 4: The number of function evaluations solving optimization problems described in Table 1 and using different algorithms

			Cons.	`DIRECT-GL-L1`			`DIRECT-L1`			`DIRECT-GLc`	`DIRECT-GLce`
#	Label	n	type	$p = 10$	$p = 10^2$	$p = 10^3$	$p = 10$	$p = 10^2$	$p = 10^3$		
1	Bunnag 1	4	L	$1,067$	$1,067$	$1,067$	$9,789$	$15,903$	$15,903$	$\mathbf{1,059}$	$7,271$
2	Bunnag 2	4	L	$5,341$	$5,341$	$5,341$	$156,317$	$> 10^6$	$> 10^6$	$\mathbf{3,663}$	$18,733$
3	Bunnag 3	5	L	$5,873$	$5,873$	$5,873$	$36,389$	$> 10^6$	$> 10^6$	$\mathbf{5,675}$	$45,483$
4	Bunnag 4	6	L	$9,433$	$12,475$	$12,531$	$8,935$	$> 10^6$	$> 10^6$	$\mathbf{5,779}$	$42,467$
5	Bunnag 5	6	L	$29,211$	$29,211$	$29,211$	$> 10^6$	$> 10^6$	$> 10^6$	$\mathbf{23,079}$	$91,445$
6	Bunnag 6	10	L	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$\mathbf{567,027}$
7	Bunnag 7	10	L	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$\mathbf{60,775}$
8	G01	13	L	11^a	11^a	11^a	7^a	7^a	7^a	$> 10^6$	$\mathbf{787,405}$
9	G02	20	NL	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$
10	G04	5	NL	43^a	43^a	$1,799$	33^a	33^a	$\mathbf{675}$	$5,907$	$21,355$
11	G06	2	NL	75^a	119^a	289^a	51^a	97^a	297^a	$\mathbf{3,461}$	$6,017$
12	G07	10	NL	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$
13	G08	2	NL	179^a	$\mathbf{471}$	$\mathbf{471}$	327^a	589	589	$\mathbf{471}$	$1,507$
14	G09	7	NL	$70,935^a$	$136,009$	$88,995$	10^6	10^6	10^6	$\mathbf{40,879}$	$89,301$
15	G10	8	NL	11^a	11^a	11^a	57^a	57^a	205^a	$> 10^6$	$\mathbf{561,857}$
16	G12*	3	NL	$\mathbf{85}$	$\mathbf{85}$	$\mathbf{85}$	111	111	123	$\mathbf{85}$	$\mathbf{85}$
17	G16	5	NL	$154,361$	$153,101$	$155,553$	$> 10^6$	$> 10^6$	$> 10^6$	$\mathbf{129,901}$	$183,779$
18	G18	9	NL	$116,767$	$120,457$	$120,481$	$334,065$	$\mathbf{105,881}$	$291,835$	$449,643$	$381,387$
19	G19	15	NL	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$
20	G24	2	NL	$1,277$	$1,277$	$1,277$	$7,865$	$140,241$	$> 10^6$	$\mathbf{709}$	$2,963$
21	Genocop 9	4	L	27^a	27^a	27^a	13^a	13^a	13^a	$\mathbf{3,191}$	$11,583$
22	Genocop 10	4	L	$4,515$	$4,509$	$4,509$	$14,093$	$> 10^6$	$> 10^6$	$\mathbf{4,331}$	$26,293$
23	Genocop 11	4	L	$49,811$	$52,145$	$52,153$	$> 10^6$	$> 10^6$	$> 10^6$	$\mathbf{41,351}$	$467,887$
24	Gold.&Price	2	NL	135^a	447	487	119^a	$\mathbf{447}$	$1,809$	$\mathbf{441}$	$2,765$
25	Himmelblau	5	NL	95^a	95^a	$4,525^a$	67^a	67^a	$3,243^a$	$\mathbf{5,305}$	$22,835$
26	Horst 1	2	L	$\mathbf{789}$	$1,051$	$1,051$	287^a	$3,689$	$273,019$	967	$4,169$
27	Horst 2	2	L	437^a	703	703	265^a	$10,829$	$> 10^6$	$\mathbf{433}$	$2,625$

Continued on next page

Table 4 Continued from previous page

			Cons.	DIRECT-GL-L1			DIRECT-L1			DIRECT-GLc	DIRECT-GLce
#	Label	n	type	$p = 10$	$p = 10^2$	$p = 10^3$	$p = 10$	$p = 10^2$	$p = 10^3$		
28	Horst 3	2	L	495	495	495	**289**	**289**	**289**	495	495
29	Horst 4	3	L	2,201	2,809	2,809	33,101	$> 10^6$	$> 10^6$	**2,021**	7,535
30	Horst 5	3	L	1,695^a	3,013	3,761	4,503^a	$> 10^6$	$> 10^6$	**2,041**	7,263
31	Horst 6	3	L	543^a	4,195	11,251	333^a	9,351^a	$> 10^6$	**4,085**	11,215
32	Horst 7	3	L	1,213	1,677	1,677	**581**	12,341	$> 10^6$	1,129	7,931
33	hs021	2	L	125	125	125	**89**	**89**	**89**	125	125
34	hs021mod	7	L	11^a	$> 10^6$	$> 10^6$	**7**	$> 10^6$	$> 10^6$	$> 10^6$	344,979
35	hs024	2	L	581	837	837	7^a	7^a	7^a	**555**	2,813
36	hs035	3	L	2,027	2,027	2,027	1,529	1,495	**1,463**	1,929	6,473
37	hs036	3	L	1,443	1,443	1,443	**727**	**727**	**727**	1,443	1,443
38	hs037	3	L	11^a	$> 11^a$	**963**	7^a	7^a	7^a	**739**	7,179
39	hs038	4	L	9,417	4,301	**4,283**	7,401	5,885	5,557	8,867	8,875
40	hs044	4	L	20,845	27,017	59,485	138,947	$> 10^6$	$> 10^6$	5,047	26,065
41	hs076	4	L	8,929	8,935	8,935	30,037	149,679	155,061	**3,509**	15,763
42	s224	2	L	295^a	943^a	737	7^a	333	**223**	823	1,309
43	s231	2	L	337	337	337	999	1,029	1,003	**331**	**331**
44	s232	2	L	11^a	75^a	1,145	19^a	57^a	$> 10^6$	**1,069**	5,601
45	s250	3	L	33^a	75^a	**2,651**	25^a	49^a	9,431	3,891	7,333
46	s251	3	L	11^a	11^a	963	7^a	7^a	$> 10^6$	**733**	7,101
47	T1	2	NL	**1,221**	1,921	1,921	3,345	8,229	8,229	1,373	2,933
48	T1	3	NL	75,105	16,625	16,333	66,137	$> 10^6$	$> 10^6$	26,643	**8,297**
49	T1	4	NL	180,383	189,595	277,587	127,087	$> 10^6$	$> 10^6$	192,951	**47,431**
50	T1	5	NL	310,195^a	520,803	616,925	$> 10^6$	$> 10^6$	$> 10^6$	253,805	**78,257**
51	T1	6	NL	394,497	708,017	698,917	$> 10^6$	$> 10^6$	$> 10^6$	239,697	**135,843**
52	T1	7	NL	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	**221,603**
53	T1	8	NL	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	**206,365**
54	T1	9	NL	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	**370,913**
55	T1	10	NL	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	**635,847**
56	zecevic2	2	L	**545**	815	815	1,533	2,961	7,079	1,081	2,763

Continued on next page

Table 4 Continued from previous page

#	Label	n	Cons. type	DIRECT-GL-L1 $p=10$	DIRECT-GL-L1 $p=10^2$	DIRECT-GL-L1 $p=10^3$	DIRECT-L1 $p=10$	DIRECT-L1 $p=10^2$	DIRECT-L1 $p=10^3$	DIRECT-GLc	DIRECT-GLce
	Aver.(overall)			$516,137$	$418,516$	$312,676$	$636,793$	$682,036$	$705,836$	$240,727$	$\mathbf{153,341}$
	Aver.($n \leq 3$)			$443,498$	$241,614$	$42,175$	$526,485$	$409,504$	$494,361$	$\mathbf{2,283}$	$4,331$
	Aver.($n \geq 4$)			$580,337$	$547,705$	$520,763$	$705,260$	$879,914$	$853,840$	$433,021$	$\mathbf{273,510}$
	Aver.(LP cons.)			$398,612$	$308,194$	$158,096$	$528,508$	$612,280$	$650,601$	$125,135$	$\mathbf{78,962}$
	Aver.(NLP cons.)			$692,335$	$558,644$	$520,906$	$764,540$	$752,595$	$754,704$	$406,577$	$\mathbf{260,058}$
	# unsolved (total)			28	21	15	34	37	38	12	**3**
	# unsolved (infes.sol.)			19	11	5	17	12	7	**0**	**0**
	# unsolved ($> 10^6$)			9	10	10	17	25	31	12	**3**

a – the final solution lies outside the feasible region

Concluded

Table 5: The number of function evaluations needed by algorithms to find a feasible point

#	Label	n	$m+r$	a	DIRECT-GLce $\varphi(\mathbf{x})$	DIRECT-GLce $\varphi^N(\mathbf{x})$	DIRECT-GL-L1 $p=10$	DIRECT-GL-L1 $p=10^2$	DIRECT-GL-L1 $p=10^3$	DIRECT-L1 $p=10$	DIRECT-L1 $p=10^2$	DIRECT-L1 $p=10^3$
8	G01	13	9	0.0111%	$\mathbf{4,050}$	$4,270$	$4,340$	$4,036$	$4,340$	$4,626$	$4,244$	$4,776$
11	G06	2	2	0.0066%	$\mathbf{102}$	$\mathbf{102}$	$1,431$	575	122	$1,521$	547	112
12	G07	10	8	0.0003%	927	$1,628$	847	$1,318$	$1,660$	$\mathbf{449}$	531	813
15	G10	8	6	0.0010%	$3,394$	$\mathbf{1,813}$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$
1e	G03	10	1	0.0000%	$\mathbf{1,381}$	$\mathbf{1,381}$	$4,037$	$3,393$	$1,413$	$> 10^6$	$> 10^6$	$> 10^6$
2e	G05	4	5	0.0000%	$6,329$	$5,658$	$8,635$	$\mathbf{5,507}$	$6,331$	$> 10^6$	$> 10^6$	$> 10^6$

a is the estimated ratio between the feasible region and the search space.

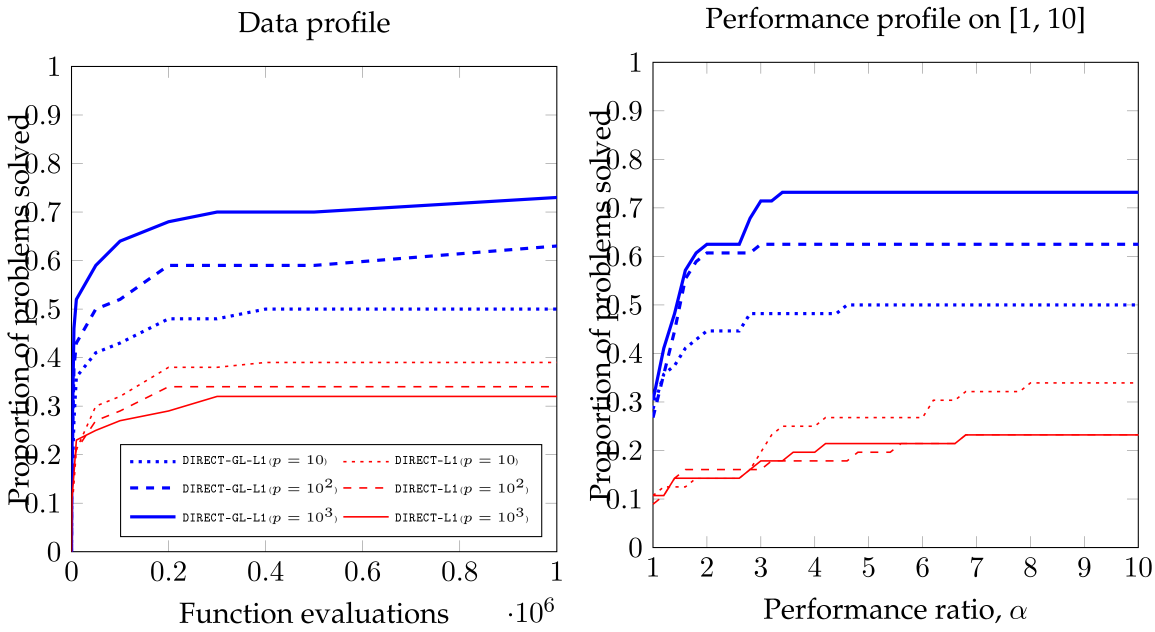

Figure 2: Data profiles (*left*) and performance profiles (*right*) of `DIRECT-GL-L1` and `DIRECT-L1` algorithms on the whole set of optimization problems described in Table 1

Experimental results are presented in Table 4 (the best results are given in bold). Here, in the second column (Label) we report the name of the problem, while in the third one – the dimensionality (n) of the problem. In the fourth column (Cons. type) we specify type of constraints: linear (L) or nonlinear (NL). Next, in the consecutive columns the total number of function evaluations are reported using four different algorithms: `DIRECT-GL-L1`, `DIRECT-L1`, `DIRECT-GLc`, and `DIRECT-GLce`, accordingly. Note, that the `DIRECT-GLc` and `DIRECT-GLce` algorithms are extensions of the `DIRECT-GL-L1` algorithm and fully described in Section 3.3.

The exact L1 penalty function approach integrated within `DIRECT-GL` (`DIRECT-GL-L1` algorithm) gives on average (Aver.(overall)) significantly better results compared to `DIRECT-L1`. However, none of tested fixed penalty parameters for L1 penalty function can ensure the convergence to the feasible solution for all tested problems. Contrary to `DIRECT-L1` which works better using smaller penalty parameters ($p = 10$), the better performance of `DIRECT-GL-L1` is achieved when larger penalty parameter values are used. When larger penalty values ($p = 10^3$) are used the `DIRECT-L1` algorithm fails for 67.9% (38/56) cases, while `DIRECT-GL-L1` fails only for 28.6% (16/56) cases accordingly. Also, larger penalty parameter values reduce the chance of obtaining a solution from the infeasible region. On the other hand, larger penalty values can bias the algorithm away from the boundary of the feasible region where the solution is often located.

Another important feature, that even for low-dimensional test problems ($n \leq 3$) `DIRECT-L1` with ($p = 10^3$) fails for 36% (9/25) cases, but the `DIRECT-GL-L1` algorithm have none such cases at all. Moreover, the smallest dimensionality when `DIRECT-L1` exceeds the maximal number of function evaluation is equal to $n = 2$, while using `DIRECT-GL-L1` the lowest dimensionality when the algorithm failed to converge withing the budged is equal to $n = 7$. When solving problems with linear (L) constraints using `DIRECT-L1` the maximal number of function evaluation is exceeded for 51.5% (17/33) cases, while for `DIRECT-GL-L1` this happens for 9.1% (3/33) cases accordingly. To sum up, while the lower penalty values give a better performance for `DIRECT-L1` algorithm, larger penalty values suit better within `DIRECT-GL-L1` scheme.

We also evaluate the performance of the algorithms using performance [DM02] and data profiles [MW09] with the convergence test (10). Performance profiles designed to compare the performance of algorithms (solvers) using a performance ratio

$$r_{p,s} = \frac{t_{p,s}}{\min\{t_{p,s} : s \in \mathcal{S}\}}. \tag{11}$$

Here $t_{p,s} > 0$ is a performance measure (the number of function evaluations in our case) obtained for each problem p from a benchmark set $\mathcal{P}$ by an algorithm s from a set of algorithms $\mathcal{S}$. The performance profile of an algorithm $s \in \mathcal{S}$ is the fraction of problems where the performance ratio is at most α

$$\rho_s(\alpha) = \frac{1}{|\mathcal{P}|}\text{size}\{p \in \mathcal{P} : r_{p,s} \leq \alpha\}, \tag{12}$$

where $|\mathcal{P}|$ is the cardinality of $\mathcal{P}$. Thus, a performance profile is a cumulative distribution function for the performance ratio. Performance profiles seek to capture how well the algorithm performs compared to other algorithms in $\mathcal{S}$ on the set of problems from $\mathcal{P}$. Algorithms with high values for $\rho_s(\alpha)$ are preferable. On the other hand, performance profiles do not provide the percentage of problems that can be solved with a given budget of function evaluations. The data profiles are designed to provide this information. The data profile defined in a such way

$$d_s(\alpha) = \frac{1}{|\mathcal{P}|}\text{size}\{p \in \mathcal{P} : t_{p,s} \leq \alpha\}, \tag{13}$$

shows the percentage of problems that can be solved with α function evaluations.

Figure 2 shows the performance and data profiles of `DIRECT-GL-L1` and `DIRECT-L1` algorithms on the whole set of optimization problems described in Table 1. The data profiles show that `DIRECT-GL-L1` algorithm outperforms `DIRECT-L1` with all penalty parameter values for all sizes of the computational budget. Moreover, the performance differences between the `DIRECT-GL-L1` and `DIRECT-L1` algorithms tend to be larger when the computational budget is bigger. The performance profiles reveal, that all three `DIRECT-GL-L1` algorithm variations solve $\approx 30\%$ with the best efficiency, while only $\approx 10\%$ using any of `DIRECT-L1` variations. `DIRECT-GL-L1` guarantees quite better performances in terms of solved problems and number of function evaluations, and that these performances are improved by combining the `DIRECT-L1` algorithm with two-step selection of potentially optimal candidates.

3.3 `DIRECT-GLce` algorithm for generally constrained global optimization problems

3.3.1 Handle the case with infeasible initial regions

In this section, we present a new way to handle hyper-rectangles with infeasible centers. In the first extension of `DIRECT-GL-L1`, we consider a situation when initial sampling points are infeasible and finding at least one feasible point can be costly. In such a situation `DIRECT`-type algorithms: `DIRECT-GL-L1` and `DIRECT-L1` are likely to fail in finding feasible points in a reasonable number of function evolutions. For such a situation we employ an additional procedure into `DIRECT-GL-L1` scheme, which samples the search space and minimizes not the original objective function, but the sum of constraint violations, i.e.:

$$\min_{\mathbf{x} \in D} \varphi(\mathbf{x}), \tag{14}$$

where

$$\varphi(\mathbf{x}) = \sum_{i=1}^{m} \max\{p_i g_i(\mathbf{x}), 0\} + \sum_{i=1}^{r} p_{i+m} |h_i(\mathbf{x})|, \tag{15}$$

until a feasible point $\mathbf{x} \in D^{\text{feas}}_{\varepsilon_\varphi}$ is found, where

$$D^{\text{feas}}_{\varepsilon_\varphi} = \{\mathbf{x} : 0 \leq \varphi(\mathbf{x}) \leq \varepsilon_\varphi, \mathbf{x} \in D\}. \tag{16}$$

Penalty parameters p_i are simply set to 1 and ε_φ is a very small acceptable constraint violation. The authors of the `eDIRECT-C` algorithm use a very similar idea, but for treating the constraints equally, they recommend to normalize every constraint function. And in the same step they sample the search space and minimize the sum of normalized constraint violations $\varphi^N(\mathbf{x})$, i.e.,

$$\min_{\mathbf{x} \in D} \varphi^N(\mathbf{x}). \tag{17}$$

In Table 5 we present the impact of this procedure on the selected subset of test problems (from Tables 12 and 13) having a small feasible region. For problems G03, G05, G10 the `L1` penalty based approaches can fail to produce a feasible solution within 10^6 function evaluations, but using (14) or (17) we avoid such a situation.

By the second extension to `DIRECT-GL-L1`, we transform problem (9) to (18):

$$\begin{aligned} &\min_{\mathbf{x} \in D} f(\mathbf{x}) + \xi(\mathbf{x}, f^{\text{feas}}_{\min}), \\ &\xi(\mathbf{x}, f^{\text{feas}}_{\min}) = \begin{cases} 0, & \mathbf{x} \in D^{\text{feas}}_{\varepsilon_\varphi} \\ \varphi(\mathbf{x}) + \Delta, & \text{otherwise,} \end{cases} \end{aligned} \tag{18}$$

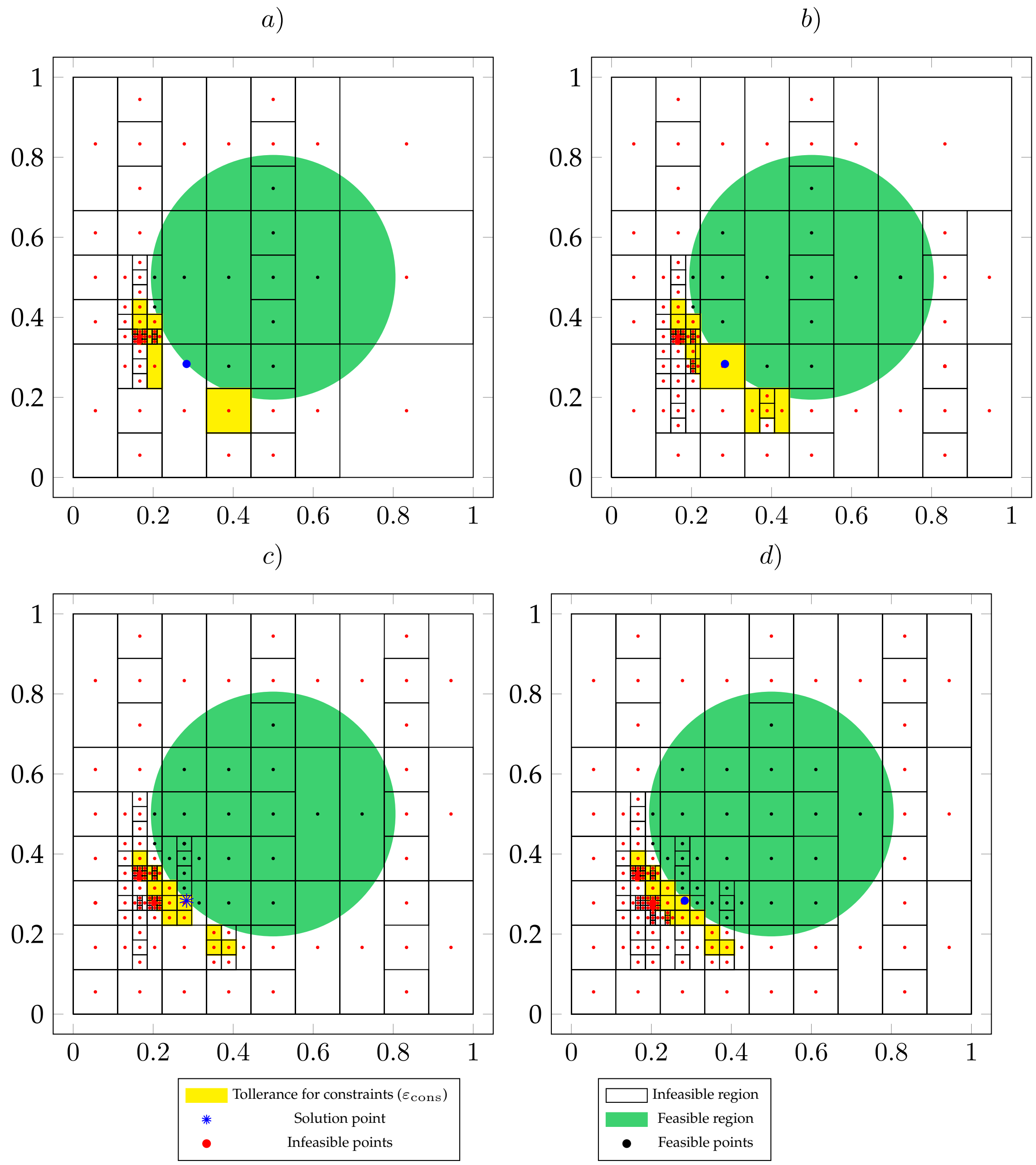

Figure 3: Geometric interpretation of `DIRECT-GLce` algorithm on T1 ($n = 2$) test problem in a) the fifth iteration, b) the sixth iteration, c) the seventh iteration, d) the eighth iteration.

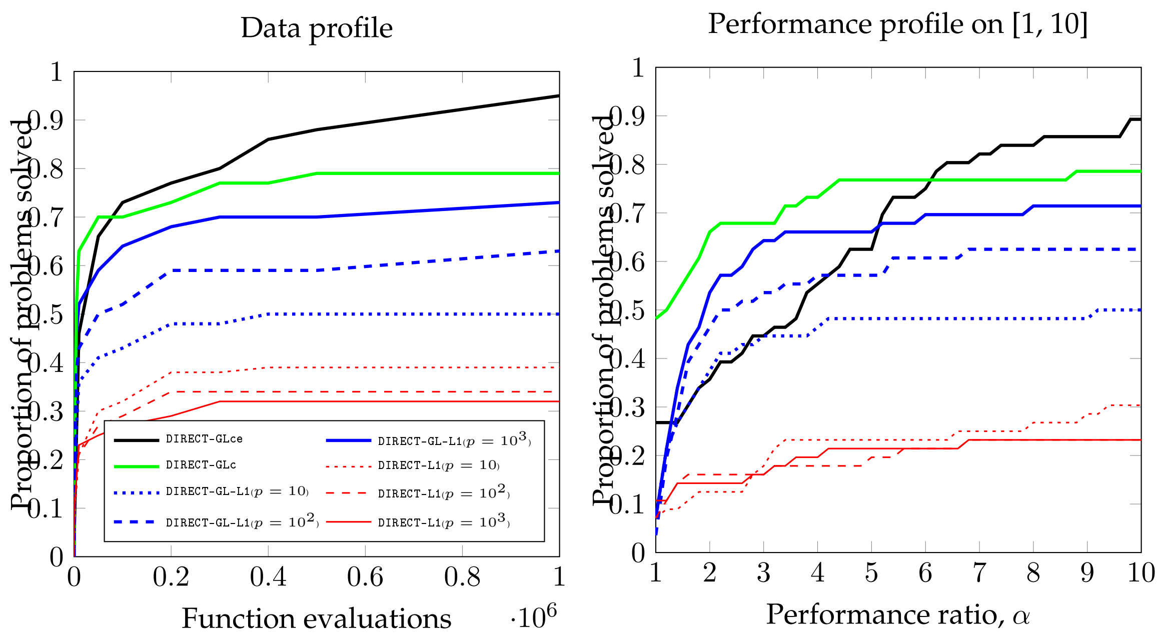

Figure 4: Data profiles (*left*) and performance profiles (*right*) of `DIRECT-GLce`, `DIRECT-GLc`, `DIRECT-GL-L1` and `DIRECT-L1` algorithms on the whole set of optimization problems described in Table 1

i.e., instead of the exact L1 penalty approach, we introduce an auxiliary function $\xi(\mathbf{x}, f_{\min}^{\text{feas}})$ which depends on the sum of constraint functions and only one parameter $\Delta = |f(\mathbf{x}) - f_{\min}^{\text{feas}}|$, which is equal to absolute value of the difference between the best feasible function value found so far $f_{\min}^{\text{feas}}$ and the objective value at an infeasible center point. The main purpose of the parameter Δ is to forbid the convergence of the algorithm to infeasible regions by penalizing objective value obtained at infeasible points. In such a way, formulation (18) does not require any penalty parameters and determine the convergence of the algorithm to a feasible solution. Note, that the value of $\xi(\mathbf{x}, f_{\min}^{\text{feas}})$ is updated during the algorithm when a smaller value of $f_{\min}^{\text{feas}}$ is found. This comes with a slight performance overhead (see Section 3.3.3 for more info on this), compared to `DIRECT-GL-L1`, which uses the fixed penalty values during the entire minimization process. The new algorithm with these two extensions is called `DIRECT-GLc`.

Note, that at the beginning of the search the difference between $f_{\min}^{\text{feas}}$ and the global solution f^* can be large, and therefore the value of $\xi(\mathbf{x}, f_{\min}^{\text{feas}})$ can be increased too much. We take into account this by modifying (18) to (19):

$$\min_{\mathbf{x}\in D} f(\mathbf{x}) + \tilde{\xi}(\mathbf{x}, f_{\min}^{\text{feas}}),$$
$$\tilde{\xi}(\mathbf{x}, f_{\min}^{\text{feas}}) = \begin{cases} 0, & \mathbf{x} \in D_{\varepsilon_\varphi}^{\text{feas}} \\ 0, & \mathbf{x} \in D_{\varepsilon_{\text{cons}}}^{\text{inf}} \\ \varphi(\mathbf{x}) + \Delta, & \text{otherwise,} \end{cases} \tag{19}$$

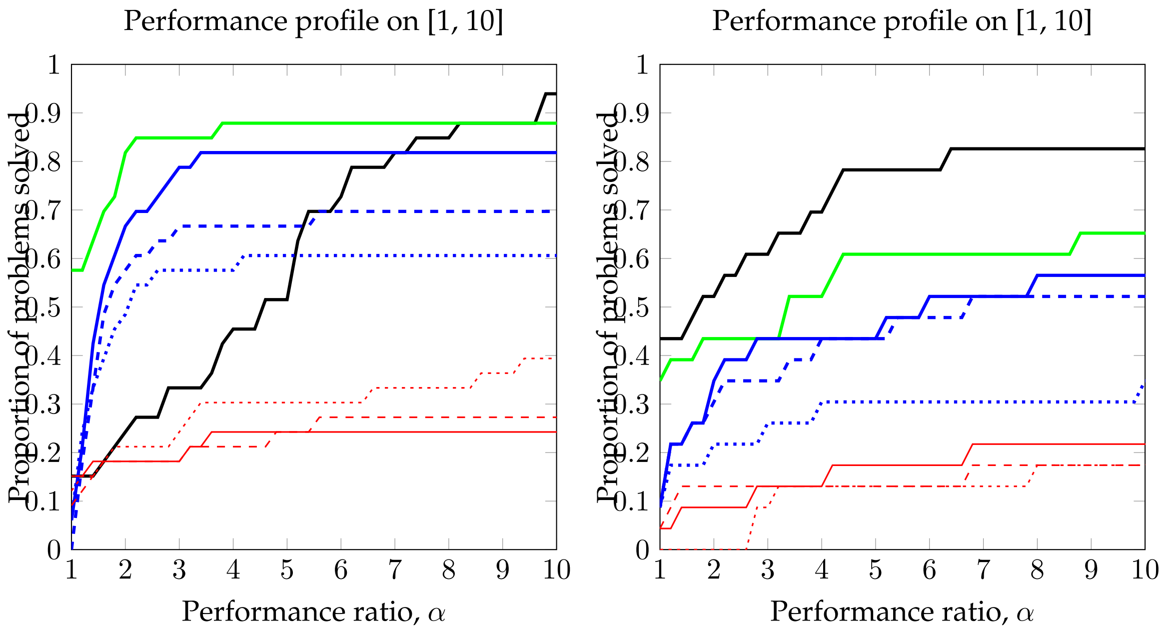

Figure 5: Performance profiles of `DIRECT-GLce`, `DIRECT-GLc`, `DIRECT-GL-L1` and `DIRECT-L1` algorithms solving problems with linear (*left*) and nonlinear (*right*) constraints from Table 1

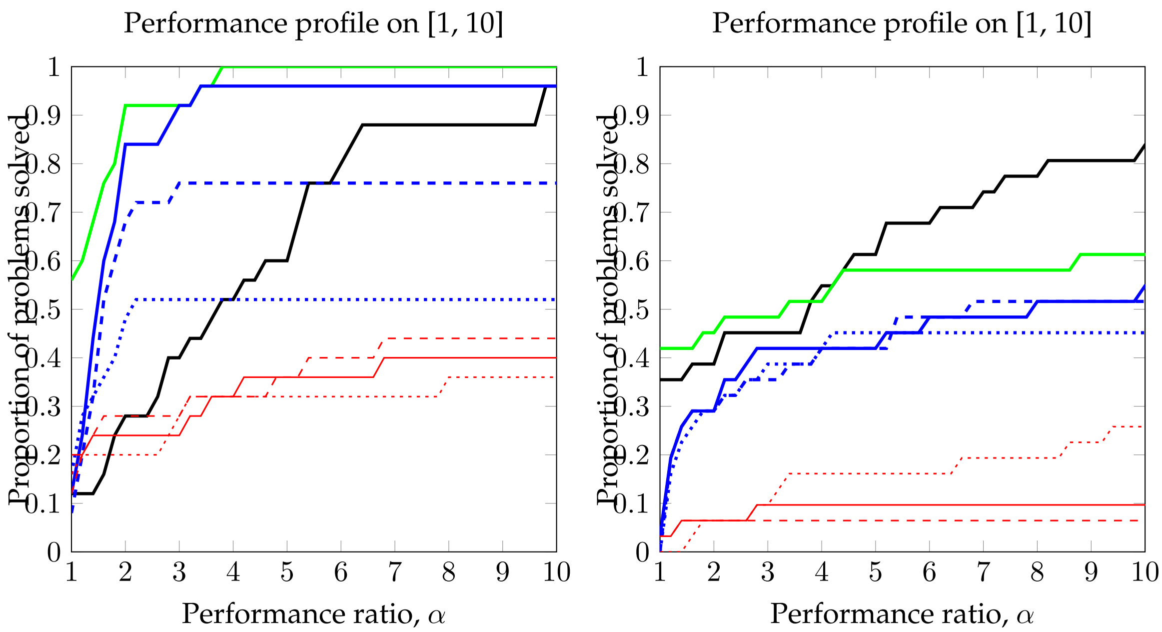

Figure 6: Performance profiles of `DIRECT-GLce`, `DIRECT-GLc`, `DIRECT-GL-L1` and `DIRECT-L1` algorithms solving $n \leq 3$ (left) and $n \geq 4$ (right) problems from Table 1

where $D^{\text{inf}}_{\varepsilon_{\text{cons}}} = \{\mathbf{x} : f(\mathbf{x}) \leq f^{\text{feas}}_{\min}, \varepsilon_{\varphi} < \varphi(\mathbf{x}) \leq \varepsilon_{\text{cons}}, \mathbf{x} \in D\}$ and $\varepsilon_{\text{cons}}$ is a small tolerance for constraint function sum, which automatically varies during the optimization process. More detailed behavior of $\varepsilon_{\text{cons}}$ is described in Algorithm 2, lines 19–28. With the introduction of this modification, the new `DIRECT-GLce` algorithm divides more hyper-rectangles with the center points lying close to the boundaries of the feasible region, i.e. potential solution. A geometrical illustration of $\varepsilon_{\text{cons}}$ parameter is shown in Fig. 3.

Experimental performance using both introduced methods are presented in Table 4. No constraint violation was allowed in this experiment and the parameter ε_{φ} was set to 0. First, it is easy to notice that for the low-dimensional test problems ($n \leq 3$) the number of function evaluations is most often smaller for `DIRECT-GLc` algorithm 46.3% (26/56), also `DIRECT-GL-L1` algorithm looks more promising with bigger penalty parameters solving the same test problems. $\varepsilon_{\text{cons}}$ parameter in `DIRECT-GLce` algorithm requires more function evaluations for simpler test problems (low dimension and with linear constrains) comparing with other algorithms, but solving more complicated test problems `DIRECT-GLce` is much more promising. The main advantage of $\varepsilon_{\text{cons}}$ parameter can be seen solving higher-dimensional and nonlinear (NL) test problems, where `DIRECT-GLce` outperforms other methods in average function evaluations and solved problems. Also looking in a general context, `DIRECT-GLce` requires less function evaluations and fails to solve only 3 test problems from which for 2 the algorithm reached the region of the global solution and only for one 20-dimensional test problem the algorithm was not able to locate the region.

Figures 4 to 6 show the data and performance profiles for all the algorithms in the interval $[1, 10]$. The data profiles from Fig. 4 display that introduced `DIRECT-GLc` and `DIRECT-GLce` algorithms significantly outperform all previously tested exact L1 penalty function based approaches, and the performance differences increase even more when the computational budget is bigger. The performance profiles in Fig. 4 reveal that `DIRECT-GLc` algorithm has the most wins and it can solve about 50% of the problems with the highest efficiency. The difference is even bigger for simpler problems (with linear constraints or $n \leq 3$), where the probability that `DIRECT-GLc` is the optimal solver is close to 0.6 (see Figs. 5 and 6). However, solving more challenging problems (with nonlinear constraints and $n \geq 4$) `DIRECT-GLce` outperforms other algorithms and the performance difference increases as the performance ratio increases. Also, if we choose being within a performance ratio of 10 of the best algorithm, then `DIRECT-GLce` is also the most effective algorithm, with the exception for simpler problems ($n \leq 3$), where `DIRECT-GLc` is the leader.

3.3.2 Algorithmic steps

The complete description of the `DIRECT-GLce` algorithm is given in Algorithm 2 and additionally is presented in a flowchart in Fig. 7. The input for the algorithm is one (or few) stopping criteria: required tolerance (ε_{pe}), the maximal number of function evaluations

input : ε_{pe}, ε_{φ}, $\mathsf{FE}_{\max}$, $\mathsf{K}_{\max}$;
output: $f_{\min}^{\text{feas}}$, $\mathbf{x}_{\min}^{\text{feas}}$, pe, k, fe;

Initialize $k = 1, fe = 1, f_{\min} = f(\mathbf{x}^1), \mathbf{x}_{\min}^k = \mathbf{x}^1, \varepsilon_{\text{cons}} = 1, \text{card}_{\text{limit}} = 10 \times n^3, \varsigma = 0, \mathbb{I}_k = \{1\}$;
if $\exists \mathbf{x}^c \in D_{\varepsilon_\varphi}^{\text{feas}}$ **then**
    Update $f_{\min}^{\text{feas}}$, $\mathbf{x}_{\min}^{\text{feas}}$ and pe;
end
while $pe > \varepsilon_{\text{pe}}$ **and** $fe < \mathsf{FE}_{\max}$ **and** $k < \mathsf{K}_{\max}$ **do** // pe defined in Eq. (7) and (21)
    if $\exists \mathbf{x}^c \in D_{\varepsilon_\varphi}^{\text{feas}}$ **then** // Phase II
        Improve the best feasible point: $S = \{f(\mathbf{x}^c) + \tilde{\xi}(\mathbf{x}^c, f_{\min}^{\text{feas}}), \mathbf{x}^c \in D, c = 1, \dots, fe\}$;
    else // Phase I
        Find feasible point: $S = \{\varphi(\mathbf{x}), \mathbf{x}^c \in D, c = 1, \dots, fe\}$;
    end
    Identify the (index) set $G_k \subseteq \mathbb{I}_k$ of POH using S in `DIRECT-GL` enhanced global search ; // Step: Selection of POH
    foreach $p \in G_k$ **do**
        Subdivide (trisect) hyper-rectangle D_k^p and update $\mathbb{I}_k$; // Step: Partitioning scheme
        Evaluate f at the centers of the new hyper-rectangles;
        Update fe;
    end
    if $\exists \mathbf{x}^c \in D_{\varepsilon_\varphi}^{\text{feas}}$ **then** // Phase II
        Update $f_{\min}^{\text{feas}}$, $\mathbf{x}_{\min}^{\text{feas}}$, $\mathbf{x}_{\min}^k$ and pe;
        if $\varepsilon_{\text{cons}} == \varepsilon_\varphi$ **and** $\varsigma \geq 10$ **then** // Control model of $\varepsilon_{\text{cons}}$
            Iteration stagnate, restart $\varepsilon_{\text{cons}} = 1$ and;
            extend limit of $\text{card}(D_{\varepsilon_{\text{cons}}}^{\text{inf}})$: $\text{card}_{\text{limit}} = \text{card}_{\text{limit}} \times 10$; // Where $\text{card}(\cdot)$ cardinality of set
        else if $D_{\varepsilon_{\text{cons}}}^{\text{inf}} ==$ **and** $\varepsilon_{\text{cons}} \times 3 \leq 10$ **then**
            Increase tolerance of constraints: $\varepsilon_{\text{cons}} = \varepsilon_{\text{cons}} \times 3$;
        else if $\text{card}(D_{\varepsilon_{\text{cons}}}^{\text{inf}}) \geq \text{card}_{\text{limit}}$ **and** $\varepsilon_{\text{cons}}/3 \geq \varepsilon_\varphi$ **then**
            Reduce tolerance of constraints: $\varepsilon_{\text{cons}} = \varepsilon_{\text{cons}}/3$;
        else if $\text{card}(D_{\varepsilon_{\text{cons}}}^{\text{inf}}) \geq \text{card}_{\text{limit}}$ **and** $\varepsilon_{\text{cons}}/3 \leq \varepsilon_\varphi$ **then**
            Set tolerance of constraints: $\varepsilon_{\text{cons}} = \varepsilon_\varphi$;
        end
    else // Phase I
        Update $\mathbf{x}_{\min}^k$;
    end
    if $\|\mathbf{x}_{\min}^k - \mathbf{x}_{\min}^{k-1}\| \geq 10^{-6}$ **then**
        Calculate distances $d(\mathbf{x}_{\min}^k, \mathbf{x}^c), \mathbf{x}^c \in D, c = 1, \dots, fe$;
        $\varsigma = 0$;
    else
        Calculate distances $d(\mathbf{x}_{\min}^k, \mathbf{x}^c), \mathbf{x}^c \in D, c = fe^{\text{old}}, \dots, fe^{\text{new}}$;
        $\varsigma = \varsigma + 1$;
    end
    $E = \{d(\mathbf{x}_{\min}^k, \mathbf{x}^c), \mathbf{x}^c \in D, c = 1, \dots, fe\}$;
    Identify the (index) set $L_k \subseteq \mathbb{I}_k$ of POH using E in `DIRECT-GL` enhanced local search ; // Step: Selection of POH
    foreach $p \in L_k$ **do**
        Subdivide (trisect) hyper-rectangle D_k^p and update $\mathbb{I}_k$; // Step: Partitioning scheme
        Evaluate f at the centers of the new hyper-rectangles;
        Update fe;
    end
    if $\exists \mathbf{x}^c \in D_{\varepsilon_\varphi}^{\text{feas}}$ **then** // Phase II
        Update $f_{\min}^{\text{feas}}$, $\mathbf{x}_{\min}^{\text{feas}}$, pe, $\mathbf{x}_{\min}^k$ and increase $k = k + 1$;
    else // Phase I
        Update $\mathbf{x}_{\min}^k$ and increase $k = k + 1$;
    end
end
return $f_{\min}^{\text{feas}}$, $\mathbf{x}_{\min}^{\text{feas}}$, pe, k, fe;

Algorithm 2: Pseudo code of the `DIRECT-GLce` algorithm

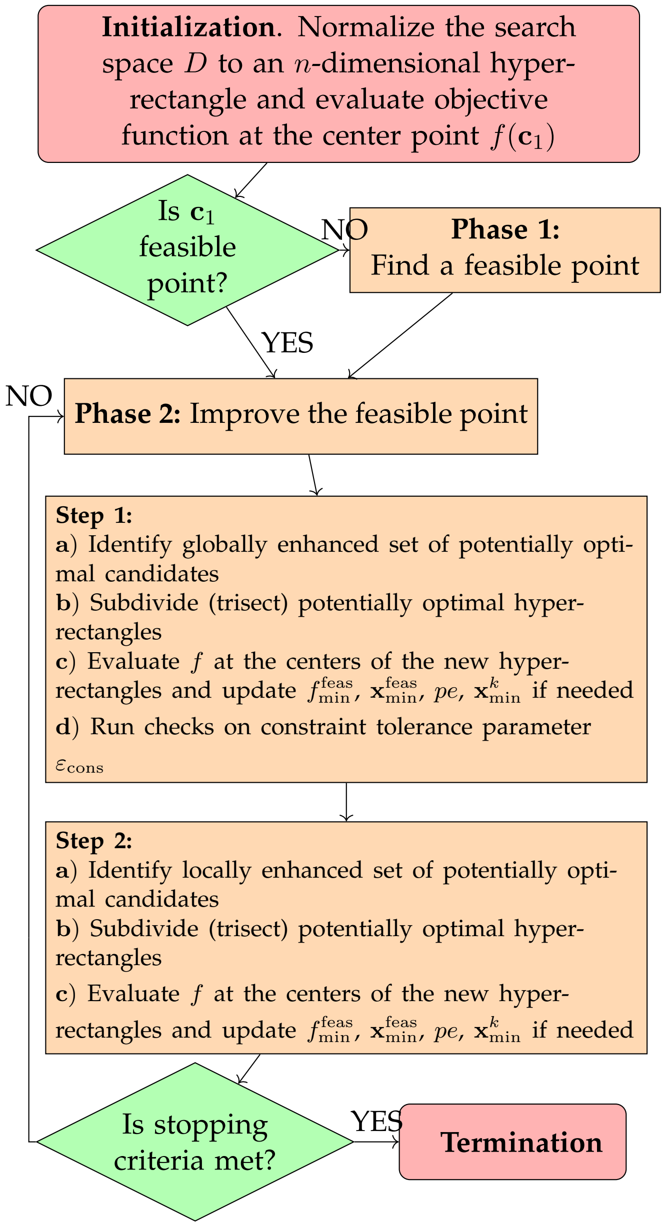

Figure 7: Flowchart of the `DIRECT-GLce` algorithm

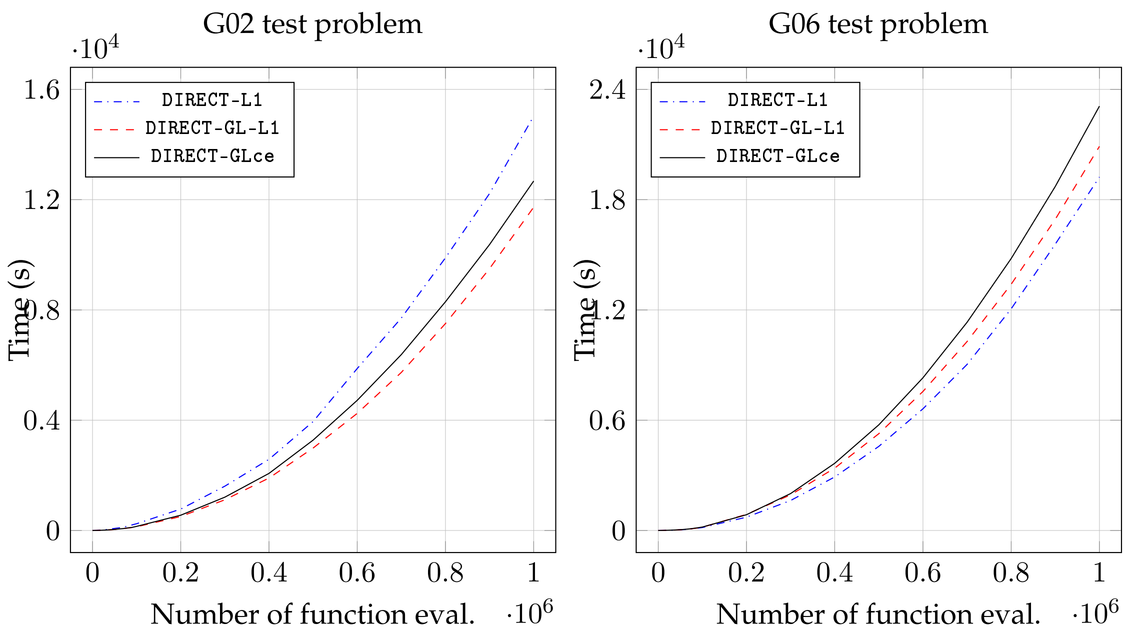

Figure 8: Geometric interpretation of running time(s) using different `DIRECT`-type methods on a few test problems

($\mathtt{FE}_{\max}$) and the maximal number of `DIRECT-GLce` iterations ($\mathtt{K}_{\max}$). After termination, `DIRECT-GLce` returns the found objective value $f_{\min}^{\text{feas}}$ and the solution point $\mathbf{x}_{\min}^{\text{feas}}$ together with algorithmic performance measures: the final tolerance – percent error (pe), the total number of function evaluations (fe), and the total number of iterations (k).

`DIRECT-GLce` uses the new two-step based strategy for the selection of potentially optimal hyper-rectangles, which is presented in [SPŽ17]. The `DIRECT-GLce` performs the selection twice in each iteration, first the globally enhanced set of potentially optimal candidates is determined and fully processed (sampled and partitioned), see Algorithm 2, lines 11–16, second the locally enhanced set is identified and fully processed, see lines 32–45.

The algorithm operates in two phases, which depends on whether a feasible point in D^{feas} is already found or not, see lines 6–10. If it is not yet found, the algorithm minimizes only sum of constraint violation (15) and attempts to find a feasible point. After such a point is found, the algorithm switches to the second phase and minimizes Problem (19). Lines 19–28 are controlled by constraint tolerance parameter $\varepsilon_{\text{cons}}$ determining infeasible points which will not be penalized at all. In the proposed strategy, the number of such points (the cardinality of the set $D_{\varepsilon_{\text{cons}}}^{\text{inf}}$), cannot exceed $10 \times n^3$, if this happens $\varepsilon_{\text{cons}}$ should be reduced. In the opposite case when the cardinality of the set $D_{\varepsilon_{\text{cons}}}^{\text{inf}}$ is zero, $\varepsilon_{\text{cons}}$ should be increased. We set the boundaries for the rate of change $10^{-4} \leq \varepsilon_{\text{cons}} \leq 10$.

3.3.3 Comment on the running time

Figure 8 shows running times of different algorithms (`DIRECT-L1`, `DIRECT-GL-L1`, `DIRECT-GLce`) from 1 to 10^6 function evaluations for G02, G06, and G19 test problems. It is observed that `DIRECT-L1` requires more running time for small dimensional test problems comparing with `DIRECT-GL-L1` and `DIRECT-GLce` algorithms. In this case the algorithm works faster than with other schemes of potential optimal hyper-rectangles selection [SPŽ17]. However, for higher dimensional test problems the proposed strategy makes the algorithm faster than `DIRECT-L1`.

3.4 Comparison with other `DIRECT`-type approaches for constrained global optimization

In this section, we present an exhaustive comparison of the newly proposed `DIRECT-GLce` algorithm with other existing `DIRECT`-type algorithms devoted to (8) problems.

3.4.1 Comparison with `eDIRECT-C` algorithm

First, we perform comparison against the recently proposed `eDIRECT-C` [LXC+17] algorithm. Authors compared their `eDIRECT-C` vs. `CORBA` [Reg14], `ConstrLMSRBF` [Reg11], `CiMPS` [KWRG11], and `DIRECT-L1` [Fin05] algorithms. The numerical experiments revealed the potential of `eDIRECT-C` algorithm for expensive constrained problems in terms of the convergence speed, the quality of final solutions and the success rate. We use two versions of `DIRECT-GLce`: the first is presented in Section 3.3, while the second version is based on `DIRECT-GLce` and is enriched with a local minimization procedure (let us call the algorithm `DIRECT-GLce-min`). To perform the comparison as fair as possible, we use the same 13 test problems from [LXC+17]. Key characteristics of these constrained global optimization test problems (G01–G13) are listed in Appendix Nr. 1., Tables 1 and 13. Note that several of these test problems: G03, G05, G11, G13 contain equality constraints, which we transform (by the same strategy as in [LXC+17]) into two inequality constraints

$$\mathbf{h}(\mathbf{x}) = 0 \rightarrow \begin{cases} \mathbf{h}(\mathbf{x}) - \varepsilon_{\mathrm{h}} & \leq 0 \\ -\mathbf{h}(\mathbf{x}) - \varepsilon_{\mathrm{h}} & \leq 0, \end{cases} \tag{20}$$

where $\varepsilon_{\mathrm{h}} > 0$ is set to 10^{-4}. The stopping criterion is the same relative error (7) as we used in the previous analysis. In these experiments allowed constraint violation $\varepsilon_{\varphi} = 0$ was used. In [LXC+17] the maximal allowed number of function evaluations was set to 1000. According to the authors, `eDIRECT-C` was developed primarily for expensive constrained global optimization problems, in which a simulation of the problem may require several hours or even days. Thus, the `eDIRECT-C` algorithm requires much more running time than the other compared methods, especially this is the case for higher dimensional problems. On the contrary, in Section 3.3 we showed that our approach

Table 6: Comparison of different algorithms for 13 test problems (see Tables 12 and 13 for the description) from [LXC+17]

#	Label	Criteria	`eDIRECT-C`	`DIRECT-GLce`	`DIRECT-GLce-min`
8	G01	$f_{\min}$	-14.9998	-14.9991	$\mathbf{-15.0000}$
		f_{eval}	148	$787,405$	$4,153$
		SR	1	–	–
9	G02	$f_{\min}$	-0.2480	-0.2246	$\mathbf{-0.3148}$
		f_{eval}	$> 1,000$	$> 10^6$	$> 10^6$
		SR	0	–	–
10	G04	$f_{\min}$	$-30,665.5385$	$-30,663.5708$	$\mathbf{-30,665.5387}$
		f_{eval}	65	$21,355$	25
		SR	1	–	–
11	G06	$f_{\min}$	$-6,961.8137$	$-6,961.1798$	$\mathbf{-6,961.8139}$
		f_{eval}	35	$6,017$	129
		SR	1	–	–
12	G07	$f_{\min}$	$\mathbf{24.3062}$	24.3332	$\mathbf{24.3062}$
		f_{eval}	152	$> 10^6$	$1,161$
		SR	1	–	–
13	G08	$f_{\min}$	$\mathbf{-0.0958}$	-0.0958	$\mathbf{-0.0958}$
		f_{eval}	154	$1,507$	115
		SR	1	–	–
14	G09	$f_{\min}$	785.6795	680.6928	$\mathbf{680.6301}$
		f_{eval}	$> 1,000$	$89,301$	41
		SR	0	–	–
15	G10	$f_{\min}$	$7,049.2484$	$7,049.8749$	$\mathbf{7,049.2480}$
		f_{eval}	105	$561,857$	3,607
		SR	1	–	–
16	G12	$f_{\min}$	$\mathbf{-1.0000}$	-0.9999	$\mathbf{-1.0000}$
		f_{eval}	52	85	17
		SR	1	–	–
1e	G03	$f_{\min}$	-0.9989^b	$\mathbf{-1.0004}$	$\mathbf{-1.0004}$
		f_{eval}	145	$251,547$	$251,547$
		SR	0	–	–
2e	G05	$f_{\min}$	$5,145.8149^b$	$5,126.5089$	$\mathbf{5,126.4967}$
		f_{eval}	413	$6,861$	$5,629$
		SR	0	–	–
3e	G11	$f_{\min}$	$\mathbf{0.7499}$	$\mathbf{0.7499}$	$\mathbf{0.7499}$
		f_{eval}	33	$1,929$	447
		SR	1	–	–
4e	G13	$f_{\min}$	0.6472	$\mathbf{0.0539}$	$\mathbf{0.0539}$
		f_{eval}	$> 1,000$	$458,239$	$100,171$
		SR	0	–	–
No. of unsolved pr.			5	2	**1**

b reported result do not satisfying the stopping criterion (7) works faster compared to DIRECT-L1 and the difference increases for larger problems. Thus, we use the maximum limit equal to 10^6 function evaluations for our algorithm. The obtained results are given in Table 6. Here, $f_{\min}$ is the minimal objective function value found by the corresponding algorithm; f_{eval} is the number of objective function evaluations required by an algorithm to reach the solution within specified accuracy; and *SR* (Success rate) records the number of success runs among the total 10 runs. Note, that our approach is deterministic and there is no requirement to run our algorithm several times.

First, observe that DIRECT-GLce algorithm solves $11/13$ of test problems while eDIRECT-C solves only $8/13$. When we combine DIRECT-GLce with the local search procedure in DIRECT-GLce-min algorithm, the hybridized algorithm outperforms eDIRECT-C by both criteria: the number solved problems $12/13$ and the quality of the final solution. Moreover, the incorporated local minimization procedure into DIRECT-GLce-min significantly reduces the total number of function evaluations compared to DIRECT-GLce, but eDIRECT-C required the smallest number of function evaluations on the average. On the other hand, authors in [LXC+17] stated that eDIRECT-C requires much more running time compared to other algorithms used in the comparison, therefore the number of function evaluations criterion alone does not represent the real performance of the algorithms very well.

Table 7: Comparison between algorithms on 20 test problems from [CRF17]

#	Label	filter-based DIRECT		DIRECT-GLc		DIRECT-GLce		DIRECT-GLce-min	
		$f_{\rm eval}$	$f_{\min}$	$f_{\rm eval}$	$f_{\min}$	$f_{\rm eval}$	$f_{\min}$	$f_{\rm eval}$	$f_{\min}$
5e	P01	25,425	0.3989	110,507	0.0294	117,367	0.0294	5,115	0.0293
6e	P02(a)	697,169	−22.4449	200,000	−397.0353	200,000	−397.1477	1,083	−400.0000
7e	P02(b)	421,197	53.6867	200,000	−397.0353	200,000	−397.1469	200,000	−400.0000
8e	P02(c)	724,337	−38.7948	200,000	−701.4834	200,000	−701.4834	1,075	−750.0000
9e	P02(d)	16,715	−399.9661	19,491	−399.9612	54,769	−399.9661	19	−400.0000
10e	P03(a)	1,109,995	−0.3832	94,197	−0.3887	117,665	−0.3887	117,665	−0.3887
11e	P05	1,009	201.1593	819	201.1593	819	201.1593	819	201.1594
12e	P09	2,203	−13.4018	1,387	−13.4018	8,271	−13.4014	71	−13.4019
13e	P12	6,665	−16.7388	23	−16.7380	23	−16.7381	5	−16.7389
14e	P13	10,583	195.3399	41,509	189.3578	41,431	189.3578	2,063	189.3466
15e	P14	1,967	−4.5140	1,695	−4.5140	9,409	−4.5139	13	−4.5142
16e	P15	105	0.0000	181	0.0000	181	0.0000	181	0.0000
17e	P16	151	0.7050	97	0.7050	97	0.7050	7	0.7049
57	P03(b)	347	−0.3889	461	−0.3887	985	−0.3887	11	−0.3888
58	P04	543	−6.6662	311	−6.6662	1,949	−6.6662	11	−6.6667
59	P06	1,323	376.3002	1,223	376.3002	1,791	376.3062	7	376.2919
60	P07	1,417	−2.8282	425	−2.8282	2,705	−2.8282	13	−2.8284
61	P08	883	−118.7010	1,197	−118.6892	1,947	−118.6898	7	−118.7052
62	P10	587	0.74183	319	0.7418	2,455	0.7418	7	0.7418
63	P11	5	−0.5000	11	−0.5000	11	−0.5000	11	−0.5000
Average		151,131		43,693		48,094		16,409	
# of unsolved		5		3		3		1	

3.4.2 Comparison with filter-based `DIRECT` algorithm

In the second part, we compare the proposed algorithms with the filter-based `DIRECT` algorithm [CRF17]. Note, that in this comparison we omit two other `DIRECT`-type algorithms based on the exact penalty functions: `EPGO`, `DF-EPGO`, as comparison with them was already carried out in [CRF17].

We consider the same 20 global optimization test problems (P01(x)–P16) see Tables 1 and 13 in Appendix Nr. 1. for the detailed description) used in [CRF17] and collected from [BFM10]. In order to provide as fair as possible comparison, in the same vein as in [CRF17] we have performed algebraic manipulation aiming to reduce the number of variables and equality constraints:

- Test problems P02(a), P02(b) and P02(c) after reformulation contain 5 variables and 10 inequality constraints. In the original problem formulation there were 9 variables, 4 equality and 2 inequality constraints.
- Test problem P02(d) after reformulation contains 5 variables and 12 inequality constraints. In the original problem formulation there were 10 variables, 5 equality and 2 inequality constraints.
- Test problem P05 after reformulation contains 2 variables, 2 equality and 2 inequality constraints. In the original problem formulation there were 3 variables and 3 equality constraints.
- Test problem P09 after reformulation contains 3 variables and 9 inequality constraints. In the original problem formulation there were 6 variables, 3 equality and 3 inequality constraints.
- Test problem P12 after reformulation contains 1 variable and 2 inequality constraints. In the original problem formulation there were 2 variables and 1 equality constraints.
- Test problem P14 after reformulation contains 3 variables and 4 inequality constraints. In the original problem formulation there were 4 variables, 1 equality and 2 inequality constraints.
- Test problem P16 after reformulation contains 2 variables and 6 inequality constraints. In the original problem formulation there were 5 variables and 3 equality constraints.

In [CRF17] authors stopped considered algorithms when the point $\bar{\mathbf{x}}$ was generated such that the percent error $(\tilde{pe})$:

$$\tilde{pe} = \frac{|f(\bar{\mathbf{x}}) - f^*|}{\max\{1, |f^*|\}} < 10^{-4}, \qquad (21)$$

Table 8: The best solutions obtained by the algorithms for problem E01

x_i, g_i	eDIRECT-C	DIRECT-GLce	DIRECT-GLce-min
x_1	3.5000	3.5003	3.5000
x_2	0.7000	0.7000	0.7000
x_3	17.0000	17.0000	17.0000
x_4	7.3000	7.3001	7.3000
x_5	7.7153^{b}	7.8000	7.8000
x_6	3.3502	3.3505	3.3502
x_7	5.2867	5.2867	5.2867
$g_1(\mathbf{x})$	-0.0739	-0.0740	-0.0739
$g_2(\mathbf{x})$	-0.1980	-0.1981	-0.1980
$g_3(\mathbf{x})$	-0.4992	-0.4992	-0.4992
$g_4(\mathbf{x})$	-0.9046	-0.9015	-0.9015
$g_5(\mathbf{x})$	-4.78×10^{-6}	-8.77×10^{-5}	-1.40×10^{-13}
$g_6(\mathbf{x})$	$2.53 \times 10^{-6\dagger}$	-7.11×10^{-5}	-3.57×10^{-14}
$g_7(\mathbf{x})$	-0.7025	-0.7025	-0.7025
$g_8(\mathbf{x})$	0.0000	-2.25×10^{-5}	-2.89×10^{-14}
$g_9(\mathbf{x})$	-0.5833	-0.5833	-0.5833
$g_{10}(\mathbf{x})$	-0.0513	-0.0513	-0.0513
$g_{11}(\mathbf{x})$	-6.48×10^{-7}	-0.0108	-0.0109
$f_{\min}$	2994.4711^{a}	2996.5498	2996.3481
f_{eval}	118	110,387	233

a – result is outside the feasible region
b – variable bound constraint violation
† – constraint is violated

or when the number of iterations exceeds the prescribed limit of 200. Note that although all considered algorithms belong to DIRECT-type class, the cost of one iteration can vary significantly. Therefore, we stopped our tested algorithms either when (21) was satisfied or when the maximal number of function evaluations equal to $200,000$ was reached. In the same vein as in [CRF17] allowed constrain violation ε_φ was set to 10^{-4}. The obtained experimental results are presented in Table 7. Our algorithms failed to locate solution point with required tolerance (21) only for $3/20$ of test problems (highlighted in red color in colored version) and none of those 3 problems was solved by filter-based DIRECT algorithm among with 2 others. Our enriched version with a local minimization procedure DIRECT-GLce-min failed only on P02(b) test problem, where the algorithm converges to a local minimum point.

3.5 Comparison on four engineering problems

In this section, we conclude our experimental investigation by applying the algorithms from the previous section to four important real-world engineering problems. The detailed description of these engineering problems can be found in [LXC+17], while in Appendix A we provide the short description and mathematical formulations. The same stopping rule (10) as in the previous section is used. No constraint violation was allowed in this experiments and the parameter ε_φ was set to 0. Note, that some of the problems contain integer variables, thus by the same analogy to [LXC+17], we regard them as continuous ones.

Tables 8 to 11 list the best found solutions and the total number of function evaluations using each of the algorithms solving four engineering problems. We note that using the `eDIRECT-C` algorithm sometimes obtained solution is better compared to ours, but in all these cases the reported solution point violates constraints of the problem. Possibly this is within constraint violation tolerances allowed by the authors of `eDIRECT-C`, but our algorithms provide final solutions without any constraint violation. As we tried to maintain the same number of decimals across the manuscript, we acknowledge that some provided rounded solution points can slightly violate constraints. For the NASA speed reducer design problem (E01) (see Table 8), the variable bounds for x_5 are $7.8 \leq x_5 \leq 8.3$, however the value of x_5 from the reported optimal solution point for `eDIRECT-C` algorithm is equal to $x_5 = 7.71532$.

A similar situation is when solving the Pressure vessel design problem (E02). The variable x_1 is bounded within $1 \leq x_1 \leq 1.375$, but the fifth constraint function $g_5(\mathbf{x}) : 1.1 - x_1 \leq 0$ reduces the feasible interval to $1.1 \leq x_1 \leq 1.375$. However, the value of x_1 for the reported optimal solution point using `eDIRECT-C` is equal to $x_1 = 1$.

Once again, we notice the similar situation solving Tension spring design problem (E03). The reported optimal solution point for `eDIRECT-C` algorithm violates the constrain $g_1(\mathbf{x}) : 1 - \frac{x_2^3 x_3}{71875 x_1^4} \leq 0$. At the solution point the feasible value of the first constraint should be non-positive, but the reported value is $g_1(\mathbf{x}) = 0.0012 > 0$.

Only in Three-bar truss design problem (E04) reported optimal solution point for `eDIRECT-C` algorithm did not violate any constraint. Our `DIRECT-GLce-min` version obtained the identical solution point. In overall view, our algorithms for all engineering problems are able to locate solution points which meet the stopping rule (7) and satisfy all the constraints.

4 Conclusions

In Section 2 we introduced a new strategy for the selection of the extended set of potentially optimal hyper-rectangles in the `DIRECT`-type algorithmic framework. Using the proposed `DIRECT-GL` approach two well-known weaknesses of `DIRECT`-type algorithms were addressed. The experimental results confirmed the well-known fact that while for simpler problems `DIRECT` performs well, for more challenging (higher dimensional) and multimodal problems the proposed modified `DIRECT-GL` performs significantly faster. Moreover, since the set of potentially optimal hyper-rectangles is larger (compared to `DIRECT`), `DIRECT-GL` scheme looks promising for more efficient parallelization too.

In Section 3, we introduced a new strategy for constrained optimization problems in

Table 9: The best solutions obtained by the algorithms for problem E02

x_i, g_i	`eDIRECT-C`	`DIRECT-GLce`	`DIRECT-GLce-min`
x_1	1.0000	1.1001	1.1000
x_2	0.6250	0.6250	0.6250
x_3	51.8135	56.9978	56.9948
x_4	84.5786	50.9916	51.0013
$g_1(\mathbf{x})$	-2.89×10^{-14}	-1.31×10^{-14}	-6.17×10^{-14}
$g_2(\mathbf{x})$	-0.1307	-0.0813	-0.0813
$g_3(\mathbf{x})$	-0.1046	-76.9749	-4.77×10^{-8}
$g_4(\mathbf{x})$	-155.4215	-189.0084	-188.9988
$g_5(\mathbf{x})$	$0.1000^{\dagger}$	-7.05×10^{-5}	-1.41×10^{-13}
$g_6(\mathbf{x})$	-0.0250	-0.0250	-0.0250
$f_{\min}$	7006.7816^{a}	7164.3701	7163.7395
f_{eval}	412	$129,097$	73

a – result is outside the feasible region
† – constraint is violated

Table 10: The best solutions obtained by the algorithms for problem E03

x_i, g_i	`eDIRECT-C`	`DIRECT-GLce`	`DIRECT-GLce-min`
x_1	0.0517	0.0518	0.0517
x_2	0.3567	0.3602	0.3569
x_3	11.2882	11.1026	11.2934
$g_1(\mathbf{x})$	$0.0012^{\dagger}$	-1.20×10^{-5}	-3.80×10^{-10}
$g_2(\mathbf{x})$	-2.61×10^{-6}	-2.73×10^{-6}	-1.68×10^{-10}
$g_3(\mathbf{x})$	-4.0568	-4.0574	-4.0510
$g_4(\mathbf{x})$	-0.7277	-0.7253	-0.7276
$f_{\min}$	0.0127^{a}	0.0127	0.0127
f_{eval}	292	$20,845$	11

a – result is outside the feasible region
† – constraint is violated

Table 11: The best solutions obtained by the algorithms for problem E04

x_i, g_i	`eDIRECT-C`	`DIRECT-GLce`	`DIRECT-GLce-min`
x_1	0.7887	0.7840	0.7887
x_2	0.4083	0.4218	0.4083
$g_1(\mathbf{x})$	-1.52×10^{-12}	-2.43×10^{-5}	-1.52×10^{-12}
$g_2(\mathbf{x})$	-1.4641	-1.4488	-1.4641
$g_3(\mathbf{x})$	-0.5359	-0.5512	-0.5359
$f_{\min}$	263.8958	263.9158	263.8958
f_{eval}	26	$21,331$	11

the DIRECT-type algorithmic framework. Two well-known weaknesses of DIRECT-L1 algorithms were addressed in the proposed approaches. First, we have demonstrated that the exact L1 penalty function based new DIRECT-GL-L1 algorithm gives on average significantly better results compared to DIRECT-L1. Moreover, the performance differences between DIRECT-GL-L1 and DIRECT-L1 algorithms tend to be larger when solving harder problems.

Next, instead of the exact L1 penalty approach, we introduced an auxiliary function based approach in the DIRECT-GLc and DIRECT-GLce algorithms, which does not require any penalty parameters. The proposed DIRECT-GLc and DIRECT-GLce algorithms significantly outperform all previously tested exact L1 penalty function based approaches, and the performance differences increases when the computational budget is larger. The DIRECT-GLc algorithm has the most wins, and it can solve about 50% of the problems with the highest efficiency. However, solving more challenging problems (with nonlinear constraints and $n \geq 4$) DIRECT-GLce outperforms other algorithms, and the performance difference increases as the performance ratio increases. Also solving higher-dimensional test problems, DIRECT-GLce outperforms the original DIRECT-L1 algorithm in running speed.

To improve the solution accuracy and improve the efficiency solving high-dimensional problems, we have enriched DIRECT-GLce with a local minimization procedure and called the new algorithm DIRECT-GLce-min. The further experimental investigation revealed the advantage of the DIRECT-GLce and DIRECT-GLce-min algorithms over most test problems and four engineering problems comparing with recent relevant approaches DIRECT-L1, filter-based DIRECT, and eDIRECT-C.

One of the most significant challenges of the partitioned based DIRECT-type approaches is dealing with optimization problems with equality constraints. Proposed DIRECT-GLce showed promising results solving such problems, but effectiveness strongly depends on the allowed equality constraints violation.

Finally, as global optimization problems are computationally expensive, one of the primary upcoming goals is to develop and investigate a parallel version of our algorithm. There are very few works devoted to the parallelization of the DIRECT-type methods. One of the primary motivations stems from the fact that the set of potentially optimal hyper-rectangles in our algorithms is larger (compared to DIRECT), thus we can expect better efficiency compared to existing parallel DIRECT-type approaches.

Data access statement

Data underlying this article can be accessed on Zenodo at https://dx.doi.org/10.5281/zenodo.1218981, and used under the Creative Commons Attribution license.

References

[BDLM12] A. Basudhar, C. Dribusch, S. Lacaze, and S. Missoum. Constrained efficient global optimization with support vector machines. *Structural and Multidisciplinary Optimization*, 46(2):201–221, 2012.

[BFM10] E. G. Birgin, C. A. Floudas, and J. M. Martínez. Global minimization using an augmented lagrangian method with variable lower-level constraints. *Mathematical Programming*, 125(1):139—-162, 2010.

[BG04] Lorenz T. Biegler and Ignacio E. Grossmann. Retrospective on optimization. *Computers & Chemical Engineering*, 28(8):1169–1192, 2004.

[BH99] Mattias Björkman and Kenneth Holmström. Global optimization using the DIRECT algorithm in Matlab. *Advanced Modeling and Optimization*, 1(2):17–37, 1999.

[BWG+00] C. A. Baker, L. T. Watson, B. Grossman, W. H. Mason, and R. T. Haftka. Parallel global aircraft configuration design space exploration. In A. Tentner, editor, *High Performance Computing Symposium 2000*, pages 54–66. Soc. for Computer Simulation Internat, 2000.

[CEC08] L. C. Cagnina, S. C. Esquivel, and C. A. Coello Coello. Solving engineering optimization problems with the simple constrained particle swarm optimizer. *Informatica (Ljubljana)*, 32(3):319–326, 2008.

[CRF17] M. F. P. Costa, A. M. A. C. Rocha, and E. M. G. P. Fernandes. Filter-based direct method for constrained global optimization. *Journal of Global Optimization*, in press, 2017.

[DM02] E. D. Dolan and J. J. Moré. Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91(2):201–213, 2002.

[Fin04] D. E. Finkel. MATLAB source code for DIRECT. `http://www4.ncsu.edu/~ctk/Finkel_Direct/`, 2004. Online; accessed: 2017-03-22.

[Fin05] D. E. Finkel. *Global Optimization with the* DIRECT *Algorithm*. PhD thesis, North Carolina State University, 2005.

[FK06] D. E. Finkel and C. T. Kelley. Additive scaling and the DIRECT algorithm. *Journal of Global Optimization*, 36(4):597–608, 2006.

[FK09] A. I. J. Forrester and A. J. Keane. Recent advances in surrogate-based optimization. *Progress in Aerospace Sciences*, 45(1):50–79, 2009.

[FL02] R. Fletcher and S. Leyffer. Nonlinear programming without a penalty function. *Mathematical Programming*, 91(2):239—-269, 2002.

[Fle87] R. Fletcher. *Practical Methods of Optimization*. John and Sons Chichester, second edition edition, 1987.

[Flo99] Christodoulos A Floudas. *Deterministic global optimization: theory, methods and applications*, volume 37 of *Nonconvex Optimization and Its Applications*. Springer US, 1999.

[Gab01] J. M. Gablonsky. *Modifications of the* DIRECT *Algorithm*. PhD thesis, North Carolina State University, 2001.

[GK01] J. M. Gablonsky and C. T. Kelley. A locally-biased form of the DIRECT algorithm. *Journal of Global Optimization*, 21(1):27–37, 2001.

[Hed05] A. Hedar. Test functions for unconstrained global optimization. http://www-optima.amp.i.kyoto-u.ac.jp/member/student/hedar/Hedar_files/TestGO.htm, 2005. Online; accessed: 2017-03-22.

[HPT95] R. Horst, P. M. Pardalos, and N. V. Thoai. *Introduction to Global Optimization*. Nonconvex Optimization and Its Application. Kluwer Academic Publishers, 1995.

[Jon01] D. R. Jones. The DIRECT global optimization algorithm. In Christodoulos A. Floudas and Panos M. Pardalos, editors, *The Encyclopedia of Optimization*, pages 431–440. Kluwer Academic Publishers, Dordrect, 2001.

[JPS93] D. R. Jones, C. D. Perttunen, and B. E. Stuckman. Lipschitzian optimization without the Lipschitz constant. *Journal of Optimization Theory and Application*, 79(1):157–181, 1993.

[KPS03] D. E. Kvasov, C. Pizzuti, and Ya. D. Sergeyev. Local tuning and partition strategies for diagonal GO methods. *Numerische Mathematik*, 94(1):93–106, 2003.

[KWRG11] M. Kazemi, G. G. Wang, S. Rahnamayan, and K. Gupta. Metamodel-based optimization for problems with expensive objective and constraint functions. *Journal of Mechanical Design*, 133(1):14505, 2011.

[LC14] Qunfeng Liu and Wanyou Cheng. A modified DIRECT algorithm with bilevel partition. *Journal of Global Optimization*, 60(3):483–499, 2014.

[LLP10a] Giampaolo Liuzzi, Stefano Lucidi, and Veronica Piccialli. A DIRECT-based approach exploiting local minimizations for the solution for large-scale global optimization problems. *Computational Optimization and Applications*, 45(2):353–375, 2010.

[LLP10b] Giampaolo Liuzzi, Stefano Lucidi, and Veronica Piccialli. A partition-based global optimization algorithm. *Journal of Global Optimization*, 48(1):113–128, 2010.

[LLP16] Giampaolo Liuzzi, Stefano Lucidi, and Veronica Piccialli. Exploiting derivative-free local searches in DIRECT-type algorithms for global optimization. *Computational Optimization and Applications*, 65:449–475, 2016.

[LXC+17] H. Liu, S. Xu, X. Chen, X. Wang, and Q. Ma. Constrained global optimization via a direct-type constraint-handling technique and an adaptive metamodeling strategy. *Structural and Multidisciplinary Optimization*, 55(1):155–177, 2017.

[LYZZ17] Qunfeng Liu, Guang Yang, Zhongzhi Zhang, and Jinping Zeng. Improving the convergence rate of the DIRECT global optimization algorithm. *Journal of Global Optimization*, 67(4):851–872, 2017.

[LZY15] Qunfeng Liu, Jinping Zeng, and Gang Yang. MrDIRECT: a multilevel robust DIRECT algorithm for global optimization problems. *Journal of Global Optimization*, 62(2):205–227, 2015.

[MPR+17] Jonas Mockus, Remigijus Paulavičius, Dainius Rusakevičius, Dmitrij Šešok, and Julius Žilinskas. Application of Reduced-set Pareto-Lipschitzian Optimization to truss optimization. *Journal of Global Optimization*, 67(1-2):425–450, 2017.

[MW09] J. J. Moré and S. M. Wild. Benchmarking derivative-free optimization algorithms. *SIAM Journal on Optimization*, 20(1):172–191, 2009.

[NLH17] J. Na, Y. Lim, and C. Han. A modified DIRECT algorithm for hidden constraints in an LNG process optimization. *Energy*, page 488–500, 2017.

[PCŽ16] Remigijus Paulavičius, Lakhdar Chiter, and Julius Žilinskas. Global optimization based on bisection of rectangles, function values at diagonals, and a set of Lipschitz constants. *Journal of Global Optimization*, (1):1–17, 2016.

[Pin96a] J. D. Pintér. *Global Optimization in Action (Continuous and Lipschitz Optimization: Algorithms, Implementations and Applications)*. Kluwer Academic Publishers, Dordrecht, 1996.

[Pin96b] János D Pintér. *Global optimization in action: continuous and Lipschitz optimization: algorithms, implementations and applications*, volume 6 of *Nonconvex Optimization and Its Applications*. Springer US, 1996.

[Piy67] S. A. Piyavskii. An algorithm for finding the absolute minimum of a function. *Theory of Optimal Solutions*, 2:13–24, 1967. in Russian.

[PLL+16] G. Di Pillo, G. Liuzzi, S. Lucidi, V. Piccialli, and F. Rinaldi. A direct-type approach for derivative-free constrained global optimization. *Computational Optimization and Applications*, 65(2):361–397, 2016.

[PLR10] G. Di Pillo, S. Lucidi, and F. Rinaldi. An approach to constrained global optimization based on exact penalty functions. *Journal of Optimization Theory and Applications*, 54(2):251–260, 2010.

[PSKŽ14] Remigijus Paulavičius, Yaroslav D. Sergeyev, Dmitri E. Kvasov, and Julius Žilinskas. Globally-biased DISIMPL algorithm for expensive global optimization. *Journal of Global Optimization*, 59(2-3):545–567, 2014.

[PŽ07] R. Paulavičius and J. Žilinskas. Analysis of different norms and corresponding Lipschitz constants for global optimization in multidimensional case. *Information Technology and Control*, 36(4):383–387, 2007.

[PŽ09] Remigijus Paulavičius and Julius Žilinskas. Global optimization using the branch-and-bound algorithm with a combination of Lipschitz bounds over simplices. *Technological and Economic Development of Economy*, 15(2):310–325, 2009.

[PŽ13] Remigijus Paulavičius and Julius Žilinskas. Simplicial Lipschitz optimization without the Lipschitz constant. *Journal of Global Optimization*, 59(1):23–40, 2013.

[PŽ14] Remigijus Paulavičius and Julius Žilinskas. *Simplicial Global Optimization*. SpringerBriefs in Optimization. Springer New York, New York, NY, 2014.

[PŽ16] Remigijus Paulavičius and Julius Žilinskas. Advantages of simplicial partitioning for Lipschitz optimization problems with linear constraints. *Optimization Letters*, 10(2):237–246, 2016.

[PŽG10] Remigijus Paulavičius, Julius Žilinskas, and Andreas Grothey. Investigation of selection strategies in branch and bound algorithm with simplicial partitions and combination of Lipschitz bounds. *Optimization Letters*, 4(2):173–183, 2010.

[Reg11] R. G. Regis. Stochastic radial basis function algorithms for large-scale optimization involving expensive black-box objective and constraint functions. *Computers and Operations Research*, 38(5):837–853, 2011.

[Reg14] R. G. Regis. Constrained optimization by radial basis function interpolation for high-dimensional expensive black-box problems with infeasible initial points. *Engineering Optimization*, 46(2):218–243, 2014.

[RL03] Tapabrata Ray and Kim Meow Liew. Society and civilization: An optimization algorithm based on the simulation of social behavior. *IEEE Transactions on Evolutionary Computation*, 7(4):386–396, 2003.

[Ser98] Yaroslav D Sergeyev. On convergence of "divide the best" global optimization algorithms. *Optimization*, 44(3):303–325, 1998.

[SHL+05] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y.P.Chen, A. Auger, and S. Tiwari. Problem definitions and evaluation criteria for the cec 2006 special session on constrained real-parameter optimization. *KanGAL*, pages 251–256, 2005.

[Shu72] B. O. Shubert. A sequential method seeking the global maximum of a function. *SIAM Journal on Numerical Analysis*, 9:379–388, 1972.

[SK06] Yaroslav D. Sergeyev and Dmitri E. Kvasov. Global search based on diagonal partitions and a set of Lipschitz constants. *SIAM Journal on Optimization*, 16(3):910–937, 2006.

[SK08] Ya. D. Sergeyev and D. E. Kvasov. *Diagonal Global Optimization Methods*. FizMatLit, Moscow, 2008. In Russian.

[SK17] Yaroslav D Sergeyev and Dmitri E Kvasov. *Deterministic Global Optimization: An Introduction to the Diagonal Approach*. SpringerBriefs in Optimization. Springer, 2017.

[SP18] Linas Stripinis and Remigijus Paulavičius. DIRECTLib – a library of global optimization problems for DIRECT-type methods, v1.1, 2018.

[SPŽ17] Linas Stripinis, Remigijus Paulavičius, and Julius Žilinskas. Improved scheme for selection of potentially optimal hyper-rectangles in direct. *Optimization Letters*, 2017.

[SS00] R. G. Strongin and Ya. D. Sergeyev. *Global Optimization with Non-Convex Constraints: Sequential and Parallel Algorithms*. Kluwer Academic Publishers, Dordrecht, 2000.

[SW10a] S. Shan and G. G. Wang. Metamodeling for high dimensional simulation-based design problems. *Journal of Mechanical Design*, 132(5):051009, 2010.

[SW10b] S. Shan and G. G. Wang. Survey of modeling and optimization strategies to solve high-dimensional design problems with computationally-expensive black-box functions. *Structural and Multidisciplinary Optimization*, 41(2):219–241, 2010.

[VV09] A.I.F. Vaz and L.N. Vicente. Pswarm: a hybrid solver for linearly constrained global derivative-free optimization. *Optimization Methods and Software*, 24(4–5):669–685, 2009.

Appendixes

Appendix A
The mathematical formulations of engineering problems

NASA speed reducer design problem [LXC+17, RL03]. Minimize the overall weight subject to constraints on bending stress of the gear teeth, surface stress, transverse deflections of the shafts and stresses in the shafts. This problem has seven design variables and eleven constraints The optimization problem is formulated as following:

$$
\begin{aligned}
\min\ f(\mathbf{x}) &= 0.7854x_1x_2^2(3.3333x_3^2 + 14.9334x_3 - 43.0934) \\
&\quad - 1.508x_1(x_6^2 + x_7^2) + 7.4777(x_6^3 + x_7^3) \\
&\quad + 0.7854(x_4x_6^2 + x_5x_7^2) \\
\text{s.t. } g_1(\mathbf{x}) &= \frac{27}{x_1x_2^2x_3} - 1 \le \mathbf{0},\ g_2(\mathbf{x}) = \frac{397.5}{x_1x_2^2x_3^2} - 1 \le \mathbf{0}, \\
g_3(\mathbf{x}) &= \frac{1.93x_4^3}{x_2x_3x_6^4} - 1 \le \mathbf{0},\ g_4(\mathbf{x}) = \frac{1.93x_5^3}{x_2x_3x_7^4} - 1 \le \mathbf{0}, \\
g_5(\mathbf{x}) &= \frac{((\frac{745x_4}{x_2x_3})^2 + 16.9 \times 10^6)^{0.5}}{110x_6^3} - 1 \le \mathbf{0}, \\
g_6(\mathbf{x}) &= \frac{((\frac{745x_5}{x_2x_3})^2 + 157.5 \times 10^6)^{0.5}}{85x_7^3} - 1 \le \mathbf{0}, \\
g_7(\mathbf{x}) &= \frac{x_2x_3}{40} - 1 \le \mathbf{0},\ g_8(\mathbf{x}) = \frac{5x_2}{x_1} - 1 \le \mathbf{0}, \\
g_9(\mathbf{x}) &= \frac{x_1}{12x_2} - 1 \le \mathbf{0},\ g_{10}(\mathbf{x}) = \frac{1.5x_6 + 1.9}{x_4} - 1 \le \mathbf{0}, \\
g_{11}(\mathbf{x}) &= \frac{1.1x_7 + 1.9}{x_5} - 1 \le \mathbf{0}
\end{aligned}
$$

where $2.6 \le x_1 \le 3.6,\ 0.7 \le x_2 \le 0.8,\ 17 \le x_3 \le 28,\ 7.3 \le x_4 \le 8.3,\ 7.8 \le x_5 \le 8.3,\ 2.9 \le x_6 \le 3.9,\ 5 \le x_7 \le 5.5$.

Pressure vessel design problem [KWRG11, LXC+17]. Minimize the total cost of material, forming and welding of a cylindrical vessel. This problem has four design variables and six constraints The optimization problem formulated as following:

$$
\begin{aligned}
\min\ f(\mathbf{x}) &= 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1^2x_4 \\
&\quad + 19.84x_1^2x_3 \\
\text{s.t. } g_1(\mathbf{x}) &= -x_1 + 0.0193x_3 \le \mathbf{0}, \\
g_2(\mathbf{x}) &= -x_2 + 0.00954x_3 \le \mathbf{0}, \\
g_3(\mathbf{x}) &= -\pi x_3^2x_4 - \frac{4}{3}\pi x_3^3 + 1296000 \le \mathbf{0}, \\
g_4(\mathbf{x}) &= x_4 - 240 \le \mathbf{0},\ g_5(\mathbf{x}) = 1.1 - x_1 \le \mathbf{0}, \\
g_6(\mathbf{x}) &= 0.6 - x_2 \le \mathbf{0}
\end{aligned}
$$

where $1 \le x_1 \le 1.375,\ 0.625 \le x_2 \le 1,\ 25 \le x_3 \le 150,\ 25 \le x_4 \le 240$.

Tension/compression spring design problem [KWRG11,LXC+17]. Minimize the weight subject to constraints on minimum deflection, shear stress, surge frequency and limits on outside diameter. This problem has three design variables and four constraints. The optimization problem formulated as following:

$$\begin{aligned}
&\min\ f(\mathbf{x}) = x_1^2 x_2(x_3+2)\\
&\text{s.t. } g_1(\mathbf{x}) = 1 - \frac{x_2^3 x_3}{71875 x_1^4} \le \mathbf{0},\\
&\quad g_2(\mathbf{x}) = \frac{x_2(4x_2 - x_1)}{12566 x_1^3(x_2 - x_1)} + \frac{2.46}{12566 x_1^2} - 1 \le \mathbf{0},\\
&\quad g_3(\mathbf{x}) = 1 - \frac{140.54 x_1}{x_3 x_2^2} \le \mathbf{0},\ g_4(\mathbf{x}) = \frac{x_1 + x_2}{1.5} - 1 \le \mathbf{0}
\end{aligned}$$

where $0.05 \le x_1 \le 0.2,\ 0.25 \le x_2 \le 1.3,\ 2 \le x_3 \le 15.$

Three-bar truss design problem [LXC+17, RL03]. Minimize the volume subject to stress constraints. This problem has two design variables and three constraints. The optimization problem formulated as following:

$$\begin{aligned}
&\min\ f(\mathbf{x}) = 100(2\sqrt{2}x_1 + x_2)\\
&\text{s.t. } g_1(\mathbf{x}) = \frac{\sqrt{2}x_1 + x_2}{\sqrt{2}x_1^2 + 2x_1x_2} 2 - 2 \le \mathbf{0},\\
&\quad g_2(\mathbf{x}) = \frac{x_2}{\sqrt{2}x_1^2 + 2x_1x_2} 2 - 2 \le \mathbf{0},\\
&\quad g_3(\mathbf{x}) = \frac{1}{x_1 + \sqrt{2}x_2} 2 - 2 \le \mathbf{0}
\end{aligned}$$

where $0 \le x_1 \le 1,\ 0 \le x_2 \le 1.$

Appendix Nr. 1.
Test problems with linear and nonlinear constraints

Table 12: Key characteristics of the optimization test problems with equality constraints

(#)	Label	Source	n	C. type	Variable bounds (D)	Optimum (f^*)
1e	G03	[LXC+17]	10	NL	$[0, 10]^n$	-1.0005
2e	G05	[LXC+17]	4	NL	$[10, 1, 200]^2 \times [-0.55, 0.55]^2$	5126.4967
3e	G11	[LXC+17]	2	NL	$[-1, 1]^n$	0.7499
4e	G13	[LXC+17]	5	NL	$[-2.3, 2.3]^2 \times [-3.2, 3.2]^3$	0.0539
5e	P01	[BFM10]	5	NL	$[-5, 5]^n$	0.0293
6e	P02(a)	[BFM10]	9	NL	$[0, 100] \times [0, 500]^8$	-400.0000
7e	P02(b)	[BFM10]	9	NL	$[0, 600] \times [0, 500]^8$	-600.0000
8e	P02(c)	[BFM10]	9	NL	$[0, 100] \times [0, 500]^8$	-750.0000
9e	P02(d)	[BFM10]	10	NL	$[0, 300]^2 \times [0, 100] \times [0, 200] \times [0, 100] \times [0, 300] \times [0, 100] \times [0, 200]^2 \times [0, 3]$	-600.0000
10e	P03(a)	[BFM10]	6	NL	$[0, 1]^4 \times [10^(- 5), 16]^2$	0.3888
11e	P05	[BFM10]	3	NL	$[0, 9.422] \times [0, 5.903] \times [0, 267.42]$	201.1600
12e	P09	[BFM10]	6	L	$[10^(- 5), 3] \times [10^(- 5), 4]^2 \times [0, 2]^2 \times [0, 6]$	-13.4020
13e	P12	[BFM10]	2	NL	$[0, 2] \times [0, 3]$	-16.7390
14e	P13	[BFM10]	3	NL	$[10^(- 5), 34] \times [10^(- 5), 17] \times [100, 300]$	189.3500
15e	P14	[BFM10]	4	L	$[10^(- 5), 3] \times [10^(- 5), 4] \times [0, 2] \times [0, 1]$	-4.51420
16e	P15	[BFM10]	3	NL	$[10^(- 5), 12.5] \times [10^(- 5), 37.5] \times [0, 50]$	0.0000
17e	P16	[BFM10]	5	L	$[0, 1.5834] \times [0, 3.625] \times [0, 1] \times [0, 3] \times [0, 4]$	0.7049

Table 13: Key characteristics of the constrained global optimization test problems

(#)	Label	Source	n	C. type	Variable bounds (D)	Optimum (f^*)
1	Bunnag 1	[VV09]	4	L	$[0,3]^n$	0.1117
2	Bunnag 2	[VV09]	4	L	$[0,4]^n$	-6.4049
3	Bunnag 3	[VV09]	5	L	$[0,3]\times[0,2]\times[0,4]\times[0,4]\times[0,2]$	-16.3657
4	Bunnag 4	[VV09]	6	L	$[0,1]^5\times[0,20]$	-213.0470
5	Bunnag 5	[VV09]	6	L	$[0,2]\times[0,8]\times[0,2]\times[0,1]\times[0,1]\times[0,2]$	-11.0000
6	Bunnag 6	[VV09]	10	L	$[0,1]^n$	-268.0146
7	Bunnag 7	[VV09]	10	L	$[0,1]^n$	-39.0000
8	G01	[LXC+17]	13	L	$[0,10]^9\times[0,100]^3\times[0,10]$	-15.0000
9	G02	[LXC+17]	20	NL	$[0,10]^n$	-0.8036
10	G04	[LXC+17]	5	NL	$[78,102]\times[33,45]\times[27,45]^3$	-30665.5386
11	G06	[LXC+17]	2	NL	$[13,100]\times[0,100]$	-6961.8138
12	G07	[LXC+17]	10	NL	$[-10,10]^n$	24.3062
13	G08	[LXC+17]	2	NL	$[0,10]^n$	-0.0958
14	G09	[LXC+17]	7	NL	$[-10,10]^n$	680.6300
15	G10	[LXC+17]	8	NL	$[100,10,000]\times[1,000,10,000]^2\times[10,1,000]^5$	7049.2480
16	G12*	[LXC+17]	3	NL	$[0.2,10]^n$	-1.0000
17	G16	[SHL+05]	5	NL	$[704.4148,906.3855]\times[68.6,288.88]\times[0,134.75]\times$ $[193,287.0966]\times[25,84.1988]$	-1.9051
18	G18	[SHL+05]	9	NL	$[0,10]^n$	-0.8660
19	G19	[SHL+05]	15	NL	$[0,10]^n$	32.6555
20	G24	[SHL+05]	2	NL	$[0,3]\times[0,4]$	-5.5080
21	Genocop 9	[VV09]	3	L	$[0,10]^n$	-2.4714
22	Genocop 10	[VV09]	4	L	$[0,3]\times[0,10]\times[0,10]\times[0,1]$	-4.5280
23	Genocop 11	[VV09]	6	L	$[0,5]\times[0,8]\times[0,5]\times[0,1]\times[0,1]\times[0,2]$	-11.0000
24	Goldstein & Price	[NLH17]	2	NL	$[-2,2]^n$	3.5389
25	Himmelblau	[CEC08]	5	NL	$[78,102]\times[33,45]\times[27,45]^3$	-31025.5602
26	Horst 1	[HPT95]	2	L	$[0,3]\times[0,2]$	-1.0625
27	Horst 2	[HPT95]	2	L	$[0,2.5]\times[0,2]$	-6.8995
28	Horst 3	[HPT95]	2	L	$[0,1]\times[0,1.5]$	-0.4444

Continued on next page

Table 13 Continued from previous page

(#)	Label	Source	n	C. type	Variable bounds (D)	Optimum (f^*)
29	Horst 4	[HPT95]	3	L	$[0.5, 2] \times [0, 3] \times [0, 2.8]$	-6.0858
30	Horst 5	[HPT95]	3	L	$[0, 1.2] \times [0, 1.2] \times [0, 1.7]$	-3.7220
31	Horst 6	[HPT95]	3	L	$[0, 6] \times [0, 5.0279] \times [0, 2.6]$	-32.5784
32	Horst 7	[HPT95]	3	L	$[0, 6] \times [0, 3] \times [0, 3]$	-52.8769
33	hs021	[VV09]	2	L	$[2, 50] \times [-50, 10]$	-99.9599
34	hs021mod	[VV09]	7	L	$[2, 50] \times [-50, 50] \times [0, 50] \times [2, 10] \times [-10, 10] \times$ $[-10, 0] \times [0, 10]$	4.0400
35	hs024	[VV09]	2	L	$[0, 5]^n$	-1.0000
36	hs035	[VV09]	3	L	$[0, 3]^n$	0.1111
37	hs036	[VV09]	3	L	$[0, 20] \times [0, 11] \times [0, 15]$	-3300.0000
38	hs037	[VV09]	3	L	$[0, 42]^n$	-3456.0000
39	hs038	[VV09]	4	L	$[-10, 10]^n$	0.0000
40	hs044	[VV09]	4	L	$[0, 5]^n$	-15.0000
41	hs076	[VV09]	4	L	$[0, 1] \times [0, 3] \times [0, 1] \times [0, 1]$	-4.6818
42	s224	[VV09]	2	L	$[0, 6] \times [0, 11]$	-304.0000
43	s231	[VV09]	2	L	$[-10, 10]^n$	0.0000
44	s232	[VV09]	2	L	$[0, 100]^n$	-1.0000
45	s250	[VV09]	3	L	$[0, 20] \times [0, 11] \times [0, 42]$	-3300.0000
46	s251	[VV09]	3	L	$[0, 42]^n$	-3456.0000
47	T1 $(n = 2)$	[Fin05]	2	NL	$[-4, 4]^n$	-3.4641
48	T1 $(n = 3)$	[Fin05]	3	NL	$[-4, 4]^n$	-4.2426
49	T1 $(n = 4)$	[Fin05]	4	NL	$[-4, 4]^n$	-4.8989
50	T1 $(n = 5)$	[Fin05]	5	NL	$[-4, 4]^n$	-5.4772
51	T1 $(n = 6)$	[Fin05]	6	NL	$[-4, 4]^n$	-6.0000
52	T1 $(n = 7)$	[Fin05]	7	NL	$[-4, 4]^n$	-6.4807
53	T1 $(n = 8)$	[Fin05]	8	NL	$[-4, 4]^n$	-6.9282
54	T1 $(n = 9)$	[Fin05]	9	NL	$[-4, 4]^n$	-7.3484
55	T1 $(n = 10)$	[Fin05]	10	NL	$[-4, 4]^n$	-7.7460
56	zecevic2	[VV09]	2	L	$[0, 10]^n$	-4.1249
57	P03(b)	[BFM10]	2	NL	$[10^(- 5), 16]^n$	0.3888
58	P04	[BFM10]	2	NL	$[0, 6] \times [0, 4]$	-6.6666

Continued on next page

Table 13 Continued from previous page

(#)	Label	Source	n	C. type	Variable bounds (D)	Optimum (f^*)
59	P06	[BFM10]	2	NL	$[0, 115.8] \times [10^{(} - 5), 30]$	376.2900
60	P07	[BFM10]	2	NL	$[-2, 2]^n$	-2.8284
61	P08	[BFM10]	2	NL	$[-8, 10] \times [0, 10]$	-118.7000
62	P10	[BFM10]	2	NL	$[0, 1]^n$	0.7417
63	P11	[BFM10]	2	NL	$[0, 1]^n$	-0.5000

Concluded