Structural and Multidisciplinary Optimization (2019) 59:2155–2175
https://doi.org/10.1007/s00158-018-2181-2

RESEARCH PAPER

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

Linas Stripinis[1] · Remigijus Paulavičius[1] · Julius Žilinskas[1]

Received: 6 June 2018 / Revised: 26 October 2018 / Accepted: 11 December 2018 / Published online: 5 January 2019

Abstract

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. In this article, 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 DIRECT-type algorithm · DIRECT-type constraint-handling · Nonconvex optimization · Derivative-free optimization

1 Introduction

Many real-world problems in engineering and applied sciences can be formulated as nonlinear programming global optimization problems (Biegler and Grossmann 2004; Floudas 1999; Pintér 1996; Shan and Wang 2010b).

In this paper, we are seeking the global solution of the general nonlinear programming problem:

$$\begin{aligned} \min_{\mathbf{x}\in D} \quad & f(\mathbf{x}) \\ \text{s.t.} \quad & \mathbf{g}(\mathbf{x}) \leq \mathbf{0}, \\ & \mathbf{h}(\mathbf{x}) = \mathbf{0}, \end{aligned} \tag{1}$$

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 $\Omega = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{g}(\mathbf{x}) \leq \mathbf{0}$ and $\mathbf{h}(\mathbf{x}) = \mathbf{0}\}$. We also assume that all functions are Lipschitz-continuous (with unknown Lipschitz constants) but can be nonlinear, nondifferentiable, and nonconvex.

The original DIRECT algorithm (Jones et al. 1993), as well as various modifications (Liu and Cheng 2014; Paulavičius et al. 2014, 2018; Paulavičius and Žilinskas 2013, 2014; Sergeyev and Kvasov 2006), 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 (Jones 2001). A few years later, the comparison of three different constraint handling strategies withing the DIRECT framework was carried out (Finkel 2005). 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 (Basudhar

Responsible Editor: Shapour Azarm

✉ Remigijus Paulavičius
remigijus.paulavicius@mii.vu.lt

Linas Stripinis
linas.stripinis@mii.vu.lt

Julius Žilinskas
julius.zilinskas@mii.vu.lt

[1] Vilnius University Institute of Data Science and Digital Technologies, Akademijos str. 4, LT-08663 Vilnius, Lithuania

et al. 2012; Costa et al. 2017; Liu et al. 2017; Pillo et al. 2010, 2016) for general engineering global optimization problems.

In this paper, we introduce the extension for general engineering optimization problems to our recently proposed DIRECT-GL (Stripinis et al. 2018) 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 algorithm 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.

The rest of the paper is organized as follows. In Section 2, we briefly review existing extensions of original DIRECT algorithm for generally constrained optimization. In Section 3, we review the existing DIRECT-type approaches based on the exact L1 penalty function schemes. The description of the new DIRECT-GLce algorithm is given in Section 4. In Sections 5 and 6, we compare our algorithm with filter-based DIRECT, EPGO, DF-EPGO, and eDIRECT-C on 80 test problems and 4 engineering instances. Finally, we conclude the paper in Section 7.

2 DIRECT-type methods for general optimization problem

In this section, we review and summarize existing DIRECT-type methods for (1) optimization problems.

The first DIRECT-type approach for problems with general constraints was presented in Jones (2001). 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.

The second DIRECT-type approach is based on the neighborhood assignment strategy (NAS) (Gablonsky 2001). 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 (Fletcher 1987). An exact L1 penalty approach is a transformation of the original constrained problem (1) to the form:

$$\min_{\mathbf{x}\in D} f(\mathbf{x}) + \sum_{i=1}^{m} \max\{p_i g_i(\mathbf{x}), 0\} + \sum_{i=1}^{r} p_{i+m}|h_i(\mathbf{x})|, \tag{2}$$

where p_i are penalty parameters. Comparison in Finkel (2005) 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 (Finkel 2005; Liu et al. 2017; Paulavičius and Žilinskas 2014, 2016). Recently, two new approaches based on penalty functions were proposed: EPGO (Pillo et al. 2010) and DF-EPGO (Pillo et al. 2016). 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 (Costa et al. 2017) 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 Fletcher and Leyffer (2002) 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 nondominated center points, and finally by those that have infeasible and dominated center points.

A metamodel-based (Forrester and Keane 2009; Shan and Wang 2010a, b) constrained DIRECT-type global optimization algorithm (eDIRECT-C) was recently also proposed in Liu et al. (2017). 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 1.

Table 1 Summary of the main algorithmic characteristics of DIRECT-type methods for (1) 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 nondominated, and one from infeasible dominated points	Uses two-step selection procedure from DIRECT-GL algorithm (Stripinis et al. 2018)
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 (Costa et al. 2017)	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

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

In Stripinis et al. (2018), 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 Hedar (2005). 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 Paulavičius and Žilinskas (2014), 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 B, Table 11. Description of all test problems used in this and subsequent section in a Matlab format is provided in the online resource (Stripinis and Paulavičius 2018). 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 Liu et al. (2017). 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 as follows:

$$pe \leq \varepsilon_{\text{pe}}, \tag{3}$$

where,

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

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

Experimental results are presented in Table 2 (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 4.

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

Table 2 The number of function evaluations solving optimization problems described in Table 11 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	**1,059**	7,271
2	Bunnag 2	4	L	5,341	5,341	5,341	156,317	$> 10^6$	$> 10^6$	**3,663**	18,733
3	Bunnag 3	5	L	5,873	5,873	5,873	36,389	$> 10^6$	$> 10^6$	**5,675**	45,483
4	Bunnag 4	6	L	9,433	12,475	12,531	8,935	$> 10^6$	$> 10^6$	**5,779**	42,467
5	Bunnag 5	6	L	29,211	29,211	29,211	$> 10^6$	$> 10^6$	$> 10^6$	**23,079**	91,445
6	Bunnag 6	10	L	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	**567,027**
7	Bunnag 7	10	L	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	**60,775**
8	G01	13	L	11^a	11^a	11^a	7^a	7^a	7^a	$> 10^6$	**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	**675**	5,907	21,355
11	G06	2	NL	75^a	119^a	289^a	51^a	97^a	297^a	**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	**471**	**471**	327^a	589	589	**471**	1,507
14	G09	7	NL	$70{,}935^a$	136,009	88,995	10^6	10^6	10^6	**40,879**	89,301
15	G10	8	NL	11^a	11^a	11^a	57^a	57^a	205^a	$> 10^6$	**561,857**
16	G12*	3	NL	**85**	**85**	**85**	111	111	123	**85**	**85**
17	G16	5	NL	154,361	153,101	155,553	$> 10^6$	$> 10^6$	$> 10^6$	**129,901**	183,779
18	G18	9	NL	116,767	120,457	120,481	334,065	**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	$> 10^6$	**709**	2,963
21	Genocop 9	4	L	27^a	27^a	27^a	13^a	13^a	13^a	**3,191**	11,583
22	Genocop 10	4	L	4,515	4,509	4,509	14,093	$> 10^6$	$> 10^6$	**4,331**	26,293
23	Genocop 11	4	L	49,811	52,145	52,153	$> 10^6$	$> 10^6$	$> 10^6$	**41,351**	467,887
24	Gold.&Price	2	NL	135^a	447	487	119^a	**447**	1,809	**441**	2,765
25	Himmelblau	5	NL	95^a	95^a	$4{,}525^a$	67^a	67^a	$3{,}243^a$	**5,305**	22,835
26	Horst 1	2	L	**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	$> 10^6$	**433**	2,625
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

Table 2 (continued)

#	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
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
Aver.(overall)				516,137	418,516	312,676	636,793	682,036	705,836	240,727	*153,341*
Aver.($n \leq 3$)				443,498	241,614	42,175	526,485	409,504	494,361	*2,283*	4,331
Aver.($n \geq 4$)				580,337	547,705	520,763	705,260	879,914	853,840	433,021	*273,510*
Aver.(L cons.)				398,612	308,194	158,096	528,508	612,280	650,601	125,135	*78,962*
Aver.(NL cons.)				692,335	558,644	520,906	764,540	752,595	754,704	406,577	*260,058*
Median				352,346	40,678	7,404	$>10^6$	$>10^6$	$>10^6$	*4,208*	13,673
# 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

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. Even more, using the DIRECT-L1 algorithm with all three different penalty parameters, the median value is more than 10^6, which means that more than half of test problems were not solve in allowed time, while much better (smaller) median values were obtained using the DIRECT-GL-L1 approach. To sum up, while lower penalty values give a better performance for the DIRECT-L1 algorithm, larger penalty values suit better within the DIRECT-GL-L1 scheme.

In recent years, performance (Dolan and Moré 2002) and data profiles (Moré and Wild 2009) have become a popular and widely used tool for benchmarking and evaluating the performance of several algorithms (solvers) when run on a large problem set. Thus, in this section, we also compare the performance of the algorithms using both these tools with the convergence test (3). Benchmark results are generated by running a certain algorithm s (from a set of algorithms $\mathcal{S}$ under consideration) for each problem p from a benchmark set $\mathcal{P}$, and recording the performance measure of interest, which could be, for example, the number of function evaluations, the computation time, the number of iterations or the memory used. In our case, the number of function evaluations criterion is used.

Performance profiles asses the overall performance of algorithms (solvers) using a performance ratio ($r_{p,s}$) as follows:

$$r_{p,s} = \frac{t_{p,s}}{\min\{t_{p,s} : s \in \mathcal{S}\}}, \tag{4}$$

where $t_{p,s} > 0$ is the number of function evaluations required to solve problem p by the algorithm s and $\min\{t_{p,s} : s \in \mathcal{S}\}$ is the smallest number of function evaluations by any algorithm on this problem. Then, the performance profile ($\rho_s(\alpha)$) of an algorithm $s \in \mathcal{S}$ is given by the cumulative distribution function for the performance ratio as follows:

$$\rho_s(\alpha) = \frac{1}{|\mathcal{P}|}\text{size}\{p \in \mathcal{P} : r_{p,s} \leq \alpha\}, \quad \alpha \geq 1, \tag{5}$$

where $|\mathcal{P}|$ is the cardinality of $\mathcal{P}$. Thus, $\rho_s(\alpha)$ is the probability for an algorithm $s \in \mathcal{S}$ that a performance ratio $r_{p,s}$ for each $p \in \mathcal{P}$ is within a factor α of the best possible ratio.

The performance profiles seek to capture how well the certain algorithm s performs compared to other algorithms in $\mathcal{S}$ on the set of problems from $\mathcal{P}$. In particular, $\rho_s(1)$ gives the fraction of the problems in $\mathcal{P}$ for which algorithm s is the winner, i.e., the best according to the $t_{p,s}$ criterion. In general, 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 is shown as follows:

$$d_s(\alpha) = \frac{1}{|\mathcal{P}|}\text{size}\{p \in \mathcal{P} : t_{p,s} \leq \alpha\}, \tag{6}$$

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

Figure 1 shows the performance and data profiles of `DIRECT-GL-L1` and `DIRECT-L1` algorithms on the whole set of optimization problems described in Table 11. 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.

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

4.1 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 of `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.,

$$\min_{\mathbf{x}\in D} \varphi(\mathbf{x}), \tag{7}$$

where,

$$\varphi(\mathbf{x}) = \sum_{i=1}^{m} \max\{p_i g_i(\mathbf{x}), 0\} + \sum_{i=1}^{r} p_{i+m}|h_i(\mathbf{x})|, \tag{8}$$

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

$$D^{\text{feas}}_{\varepsilon_\varphi} = \{\mathbf{x} : 0 \leq \varphi(\mathbf{x}) \leq \varepsilon_\varphi, \mathbf{x} \in D\}. \tag{9}$$

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

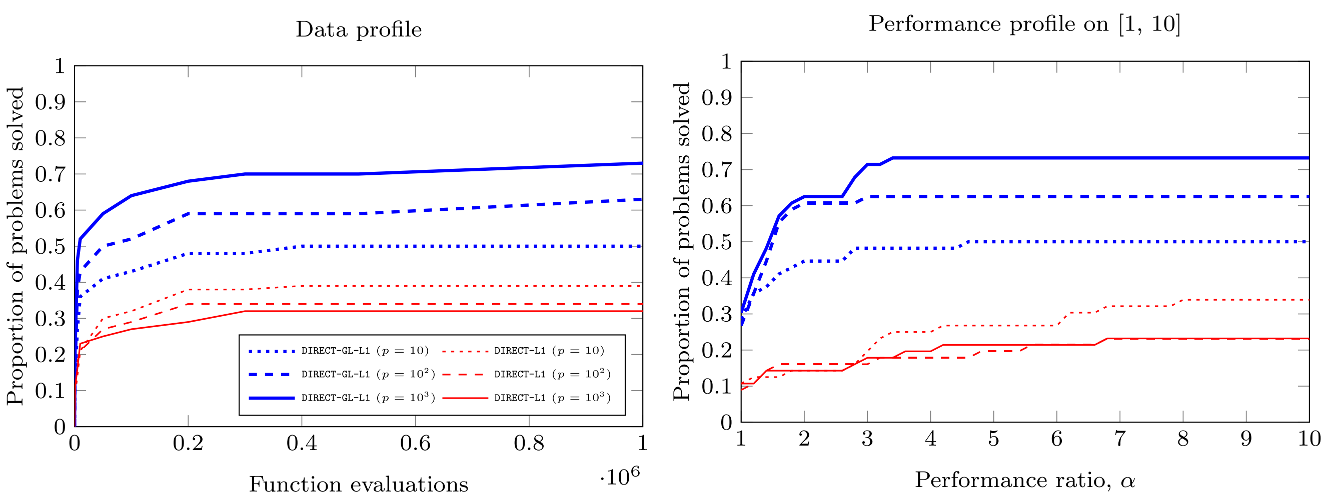

Fig. 1 Data profiles (left) and performance profiles (right) of `DIRECT-GL-L1` and `DIRECT-L1` algorithms on the whole set of optimization problems described in Table 11

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.,

$$\min_{\mathbf{x}\in D} \varphi^N(\mathbf{x}). \tag{10}$$

In Table 3, we present the impact of this procedure on the selected subset of test problems (from Tables 10 and 11) having a small feasible region (the best results are given in bold). For problems G03, G05, and G10, the `L1` penalty-based approaches can fail to produce a feasible solution within 10^6 function evaluations, but using (7) or (10), we avoid such a situation.

By the second extension to `DIRECT-GL-L1`, we transform problems (2) to (11) as follows:

$$\begin{aligned} &\min_{\mathbf{x}\in D} f(\mathbf{x}) + \xi(\mathbf{x}, f_{\min}^{\text{feas}}),\\ &\xi(\mathbf{x}, f_{\min}^{\text{feas}}) = \begin{cases} 0, & \mathbf{x} \in D_{\varepsilon_\varphi}^{\text{feas}} \\ \varphi(\mathbf{x}) + \Delta, & \text{otherwise,} \end{cases} \end{aligned} \tag{11}$$

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 (11) does not require any penalty parameters and determine the convergence of the algorithm to a feasible solution. The value of $\xi(\mathbf{x}, f_{\min}^{\text{feas}})$ is updated when a smaller value of $f_{\min}^{\text{feas}}$ is found. The new algorithm with these two extensions is called `DIRECT-GLc`. Note that this comes with a slight performance overhead, compared to `DIRECT-GL-L1`, which uses the fixed penalty values during the entire minimization process.

Since at the beginning of the search the difference between $f_{\min}^{\text{feas}}$ and the global solution f^* can be large, therefore the value of $\xi(\mathbf{x}, f_{\min}^{\text{feas}})$ can be increased too much. We take into account this by modifying (11) to (12) as follows:

$$\begin{aligned} &\min_{\mathbf{x}\in D} f(\mathbf{x}) + \tilde{\xi}(\mathbf{x}, f_{\min}^{\text{feas}}),\\ &\tilde{\xi}(\mathbf{x}, f_{\min}^{\text{feas}}) = \begin{cases} 0, & \mathbf{x} \in D_{\varepsilon_\varphi}^{\text{feas}} \\ 0, & \mathbf{x} \in D_{\varepsilon_{\text{cons}}}^{\text{inf}} \\ \varphi(\mathbf{x}) + \Delta, & \text{otherwise,} \end{cases} \end{aligned} \tag{12}$$

where $D_{\varepsilon_{\text{cons}}}^{\text{inf}} = \{\mathbf{x} : f(\mathbf{x}) \le f_{\min}^{\text{feas}}, \varepsilon_\varphi < \varphi(\mathbf{x}) \le \varepsilon_{\text{cons}}, \mathbf{x} \in D\}$ 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 1, 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. 2.

Experimental performance using both introduced methods are presented in Table 2. 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 ($n \le 3$), 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 three test problems from which for two, 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.

Table 3 The number of function evaluations needed by algorithms to find a feasible point

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

[a] is the estimated ratio between the feasible region and the search space

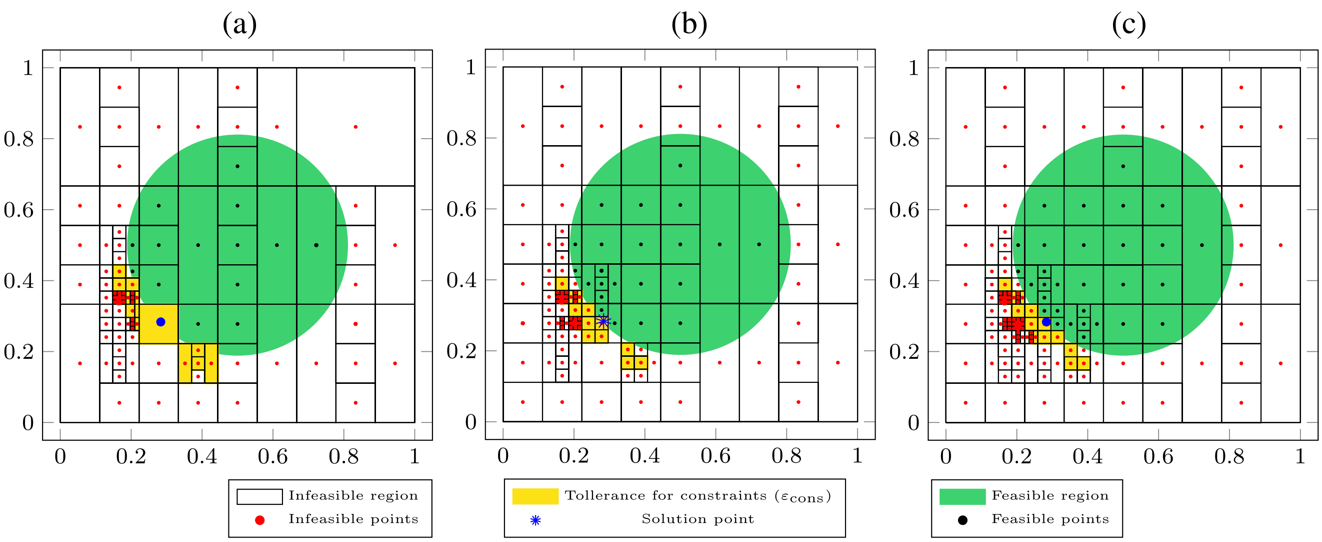

Fig. 2 Geometric interpretation of `DIRECT-GLce` algorithm on T1 ($n = 2$) test problem in (**a**) the sixth iteration, (**b**) the seventh iteration, (**c**) the eighth iteration

Figures 3, 4, and 5 show the data and performance profiles for all the algorithms in the interval [1, 10]. The data profiles from Fig. 3 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. 3 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. 4, and 5). 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.

4.2 Algorithmic steps

The complete description of the `DIRECT-GLce` algorithm is given in algorithm 1 and additionally is presented in a flowchart in Fig. 6. The input for the algorithm is one (or few) stopping criteria: required tolerance (ε_{pe}), the maximal number of function evaluations ($\mathtt{FE}_{\max}$), and the maximal number of `DIRECT-GLce` iterations ($\mathrm{K}_{\max}$). After termination, `DIRECT-GLce` returns the

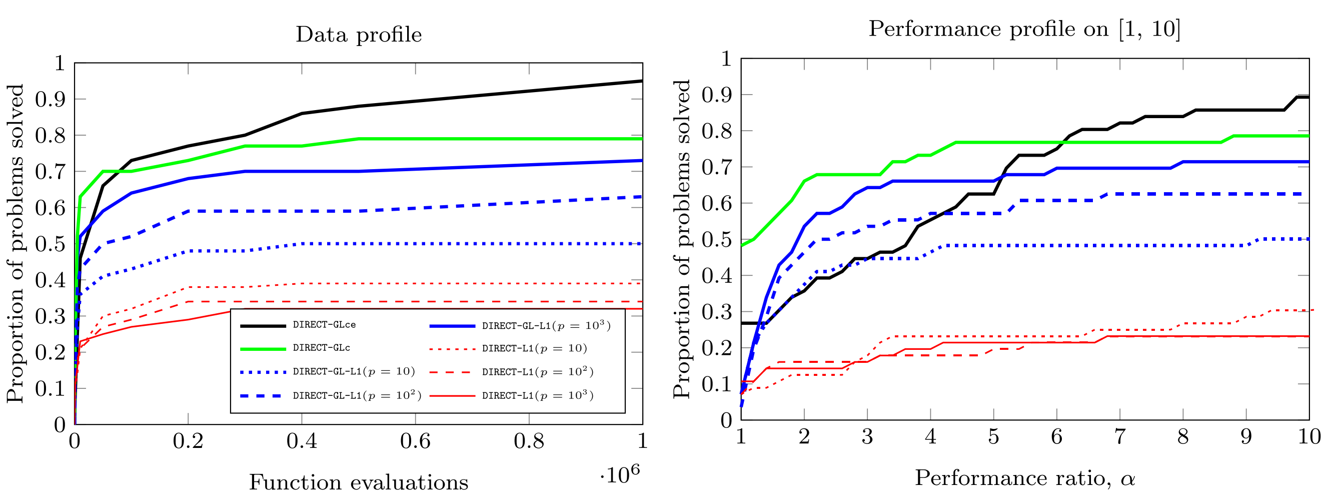

Fig. 3 Data profiles (*left*) and performance profiles (*right*) of `DIRECT-GLce`, `DIRECT-GLc`, `DIRECT-GL-L1`, and `DIRECT-L1` algorithms on the whole set of optimization problems described in Table 11

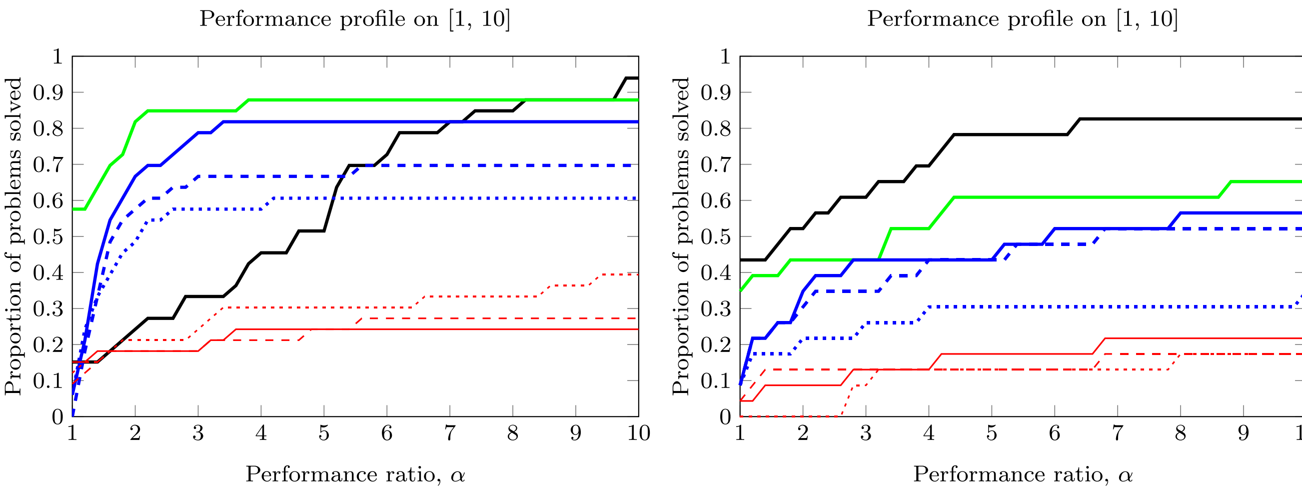

Fig. 4 Performance profiles of DIRECT-GLce, DIRECT-GLc, DIRECT-GL-L1, and DIRECT-L1 algorithms solving problems with linear (left) and nonlinear (right) constraints from Table 11

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 Stripinis et al. (2018). 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 1, 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 (8) and attempts to find a feasible point. After such a point is found, the algorithm switches to the second phase and minimizes problem (12). 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 $10^{-4} \le \varepsilon_{\text{cons}} \le 10$.

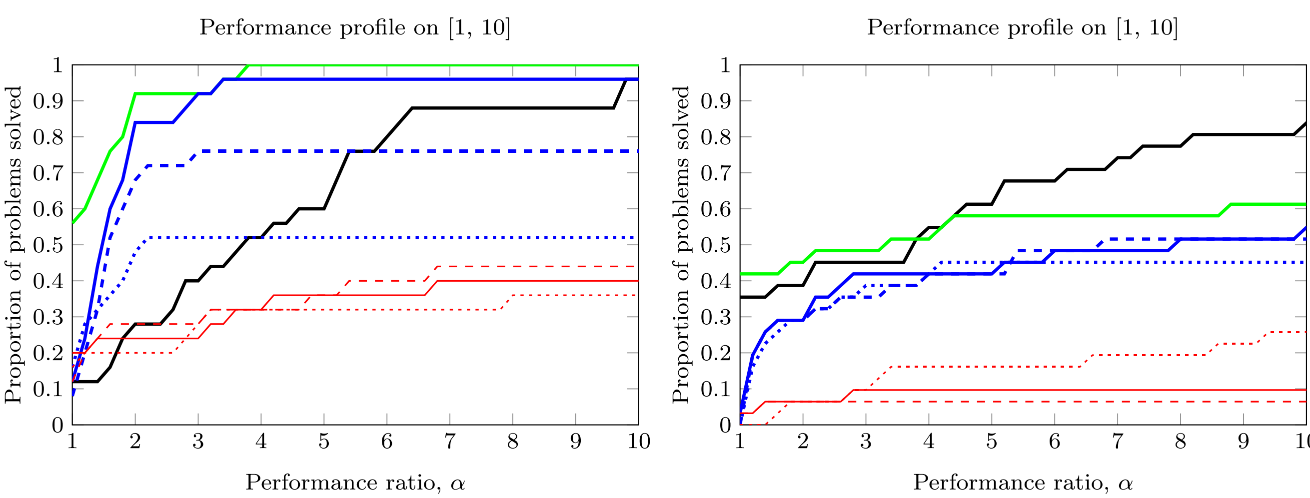

Fig. 5 Performance profiles of DIRECT-GLce, DIRECT-GLc, DIRECT-GL-L1, and DIRECT-L1 algorithms solving $n \le 3$ (left) and $n \ge 4$ (right) problems from Table 11

Algorithm 1 Pseudo code of the `DIRECT-GLce` algorithm

input : $\varepsilon_{\text{pe}}, \varepsilon_{\varphi}, \mathsf{FE}_{\max}, \mathsf{K}_{\max}$;
output: $f^{\text{feas}}_{\min}, \mathbf{x}^{\text{feas}}_{\min}, pe, k, fe$;

Initialize $k = 1$, $fe = 1$, $f_{\min} = f(\mathbf{x}^1)$, $\mathbf{x}^k_{\min} = \mathbf{x}^1$, $\varepsilon_{\text{cons}} = 1$, $\text{card}_{\text{limit}} = 10 \times n^3$, $\varsigma = 0$, $\mathbb{I}_k = \{1\}$;
if $\exists \mathbf{x}^c \in D^{\text{feas}}_{\varepsilon_\varphi}$ **then**
    Update $f^{\text{feas}}_{\min}$, $\mathbf{x}^{\text{feas}}_{\min}$ and pe;
end
while $pe > \varepsilon_{\text{pe}}$ **and** $fe < \mathsf{FE}_{\max}$ **and** $k < \mathsf{K}_{\max}$ **do** // pe defined in (3) and (14)
    if $\exists \mathbf{x}^c \in D^{\text{feas}}_{\varepsilon_\varphi}$ **then** // **Phase II**
        Improve the best feasible point: $S = \{f(\mathbf{x}^c) + \tilde{\xi}(\mathbf{x}^c, f^{\text{feas}}_{\min}), \mathbf{x}^c \in D, c = 1, \dots, fe\}$;
    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^p_k and update $\mathbb{I}_k$; // Step: Partitioning scheme
        Evaluate f at the centers of the new hyper-rectangles;
        Update fe;
    end
    if $\exists \mathbf{x}^c \in D^{\text{feas}}_{\varepsilon_\varphi}$ **then** // **Phase II**
        Update $f^{\text{feas}}_{\min}$, $\mathbf{x}^{\text{feas}}_{\min}$, $\mathbf{x}^k_{\min}$ 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 $\text{card}(D^{\inf}_{\varepsilon_{\text{cons}}})$: $\text{card}_{\text{limit}} = \text{card}_{\text{limit}} \times 10$; // Where card(·) cardinality of set
        else if $D^{\inf}_{\varepsilon_{\text{cons}}} == \emptyset$ **and** $\varepsilon_{\text{cons}} \times 3 \leq 10$ **then**
            Increase tolerance of constraints: $\varepsilon_{\text{cons}} = \varepsilon_{\text{cons}} \times 3$;
        else if $\text{card}(D^{\inf}_{\varepsilon_{\text{cons}}}) \geq \text{card}_{\text{limit}}$ **and** $\varepsilon_{\text{cons}}/3 \geq \varepsilon_\varphi$ **then**
            Reduce tolerance of constraints: $\varepsilon_{\text{cons}} = \varepsilon_{\text{cons}}/3$;
        else if $\text{card}(D^{\inf}_{\varepsilon_{\text{cons}}}) \geq \text{card}_{\text{limit}}$ **and** $\varepsilon_{\text{cons}}/3 \leq \varepsilon_\varphi$ **then**
            Set tolerance of constraints: $\varepsilon_{\text{cons}} = \varepsilon_\varphi$;
        end
    else // **Phase I**
        Update $\mathbf{x}^k_{\min}$;
    end
    if $\|\mathbf{x}^k_{\min} - \mathbf{x}^{k-1}_{\min}\| \geq 10^{-6}$ **then**
        Calculate distances $d(\mathbf{x}^k_{\min}, \mathbf{x}^c)$, $\mathbf{x}^c \in D$, $c = 1, \dots, fe$;
        $\varsigma = 0$;
    else
        Calculate distances $d(\mathbf{x}^k_{\min}, \mathbf{x}^c)$, $\mathbf{x}^c \in D$, $c = fe^{\text{old}}, \dots, fe^{\text{new}}$;
        $\varsigma = \varsigma + 1$;
    end
    $E = \{d(\mathbf{x}^k_{\min}, \mathbf{x}^c), \mathbf{x}^c \in D, c = 1, \dots, fe\}$;
    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^p_k and update $\mathbb{I}_k$; // Step: Partitioning scheme
        Evaluate f at the centers of the new hyper-rectangles;
        Update fe;
    end
    if $\exists \mathbf{x}^c \in D^{\text{feas}}_{\varepsilon_\varphi}$ **then** // **Phase II**
        Update $f^{\text{feas}}_{\min}$, $\mathbf{x}^{\text{feas}}_{\min}$, pe, $\mathbf{x}^k_{\min}$ and increase $k = k + 1$;
    else // **Phase I**
        Update $\mathbf{x}^k_{\min}$ and increase $k = k + 1$;
    end
end
return $f^{\text{feas}}_{\min}, \mathbf{x}^{\text{feas}}_{\min}, pe, k, fe$;

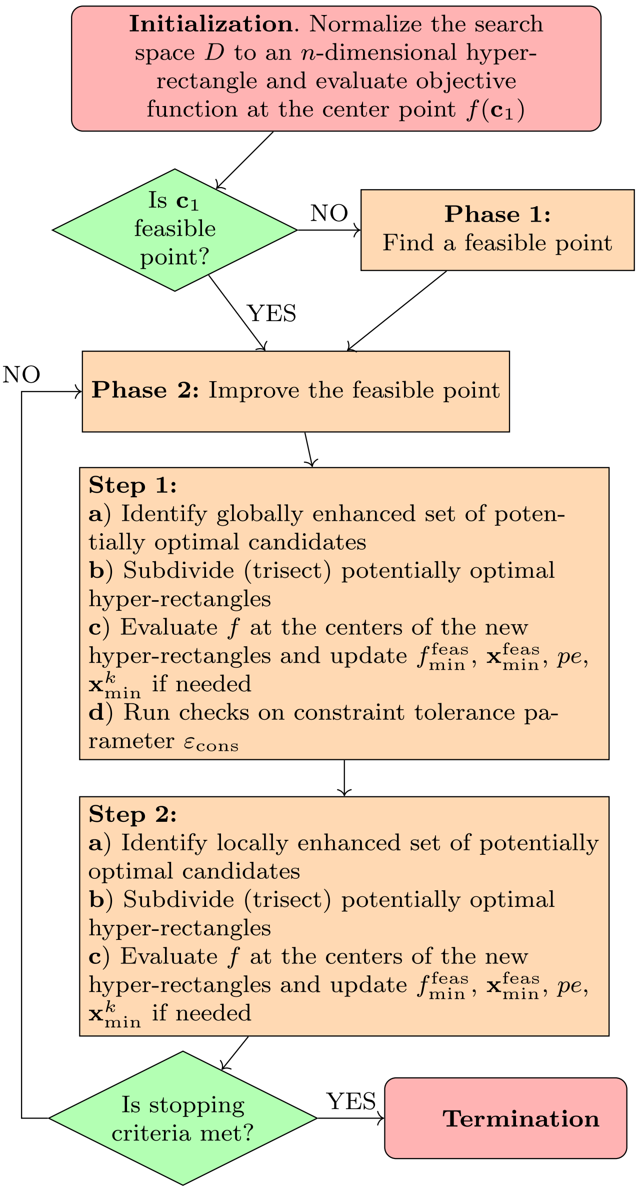

Fig. 6 Flowchart of the DIRECT-GLce algorithm

5 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 (1) problems.

5.1 Comparison with eDIRECT-C algorithm

First, we perform comparison against the recently proposed eDIRECT-C (Liu et al. 2017) algorithm. Authors compared their eDIRECT-C vs. CORBA (Regis 2014), ConstrLMSRBF (Regis 2011), CiMPS (Kazemi et al. 2011), and DIRECT-L1 (Finkel 2005) 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 4, 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 Liu et al. (2017). Key characteristics of these constrained global optimization test problems (G01–G13) are listed in Appendix B, Tables 11 and 10. Note that several of these test problems: G03, G05, G11, and G13 contain equality constraints, which we transform (by the same strategy as in Liu et al. (2017)) into two inequality constraints as follows:

$$\mathbf{h}(\mathbf{x}) = 0 \rightarrow \begin{cases} \mathbf{h}(\mathbf{x}) - \varepsilon_{\text{h}} & \leq 0 \\ -\mathbf{h}(\mathbf{x}) - \varepsilon_{\text{h}} & \leq 0, \end{cases} \tag{13}$$

where $\varepsilon_{\text{h}} > 0$ is set to 10^{-4}. The stopping criterion is the same relative error (3) as we used in the previous analysis. In these experiments, allowed constraint violation $\varepsilon_{\varphi} = 0$ was used. In Liu et al. (2017), 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 4, we showed that our approach 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 4 (as usual, the best results are given in bold). 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 ten 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

Table 4 Comparison of different algorithms for 13 test problems (see Tables 11 and 10 for the description) from Liu et al. (2017)

#	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	$f_{\min}$	**24.3062**	24.3332	**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	**680.6301**
		f_{eval}	$>1{,}000$	89,301	41
		SR	0	–	–
15	G10	$f_{\min}$	7,049.2484	7,049.8749	**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	**5,126.4967**
		f_{eval}	413	6,861	5, 629
		SR	0	–	–
3e	G11	$f_{\min}$	**0.7499**	**0.7499**	**0.7499**
		f_{eval}	33	1, 929	447
		SR	1	–	–
4e	G13	$f_{\min}$	0.6472	**0.0539**	**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 (3)

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 Liu et al. (2017) stated that eDIRECT-C requires much more running time compared

Table 5 Comparison between algorithms on 20 test problems from Costa et al. (2017)

#	Label	filter-based DIRECT		DIRECT-GLc		DIRECT-GLce		DIRECT-GLce-min	
		f_{eval}	$f_{\min}$	f_{eval}	$f_{\min}$	f_{eval}	$f_{\min}$	f_{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
Aver.(overall)		151,131		43,693		48,094		16,409	
Aver.($n \leq 3$)		1,984		3,547		5,148		230	
Aver.($n \geq 4$)		499,140		137,366		148,300		54,160	
Aver.(L cons.)		7,455		30,623		31,979		1,637	
Aver.(NL cons.)		199,023		48,049		53,465		21,333	
Median		1,692		1,210		2,580		16	
# of unsolved		5		3		3		1	

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.

5.2 Comparison with filter-based DIRECT algorithm

In the second part, we compare the proposed algorithms with the filter-based DIRECT algorithm (Costa et al. 2017). 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 Costa et al. (2017).

We consider the same 20 global optimization test problems (P01(x)–P16) see Tables 10 and 11 in Appendix B for the detailed description) used in Costa et al. (2017) and collected from Birgin et al. (2010). In order to provide as fair as possible comparison, in the same vein as in Costa et al. (2017), 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

Table 6 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}
$g_6(\mathbf{x})$	2.53×10^{-6}[c]	-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
[c]Constraint is violated

Table 7 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[b]	-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
[b]Constraint is violated

Table 8 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^b	-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

[b]Constraint is violated

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 Costa et al. (2017), the authors stopped considered algorithms when the point $\bar{\mathbf{x}}$ was generated such that the percent error ($\tilde{pe}$) was as follows:

$$\tilde{pe} = \frac{|f(\bar{\mathbf{x}}) - f^*|}{\max\{1, |f^*|\}} < 10^{-4}, \tag{14}$$

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 (14) was satisfied or when the maximal number of function evaluations equal to 200,000 was reached. In the same vein as in Costa et al. (2017) allowed constrain violation ε_φ was set to 10^{-4}. The obtained experimental results are presented in Table 5. Our algorithms (DIRECT-GLc and DIRECT-GLce) give on average (aver.(overall)) significantly better results compared to filter-based DIRECT and failed to locate solution point with required tolerance (14) only for 3/20 of test problems (highlighted in red color in the colored version), and none of those three problems was solved by the filter-based DIRECT algorithm among with two others. However, for simpler test problems, i.e., lower dimensionality cases ($n \leq 3$) and on problems with linear (L) constraints filter-based DIRECT is a very promising option. The completely different behavior for harder test problems, i.e., higher dimensionality cases ($n \geq 4$) and on problems with nonlinear (NL) constraints where our approaches give much better results. Finally, our enriched version with a local minimization procedure

Table 9 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

DIRECT-GLce-min failed only on P02(b) test problem, where the algorithm converged to a local minimum point and gave the best results based on all used comparison criteria.

6 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 Liu et al. (2017), while in Appendix A, we provide the short description and mathematical formulations. The same stopping rule (3) 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 Liu et al. (2017), we regard them as continuous ones.

Tables 6, 7, 8, and 9 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 6), 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 nonpositive, 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 (3) and satisfy all the constraints.

7 Conclusions, challenges, and further work

In this paper, we introduced a new strategy for constrained optimization problems in 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.

Acknowledgements The authors would like to thank all anonymous referees for their valuable comments and suggestions to improve the paper.

Funding information The research work of R. Paulavičius and L. Stripinis was funded by a Grant (No. P-MIP-17-60) from the Research Council of Lithuania.

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.

Appendix A: The mathematical formulations of engineering problems

NASA speed reducer design problem (Liu et al. 2017; Ray and Liew 2003). The overall weight subject to constraints on bending stress of the gear teeth, surface stress, and transverse deflections of the shafts and stresses in the shafts is minimized. This problem has 7 design variables and 11 constraints. The optimization problem is formulated as following:

$$\begin{aligned}
\min \quad & f(\mathbf{x}) = 0.7854x_1x_2^2(3.3333x_3^2+14.9334x_3-43.0934) \\
& -1.508x_1(x_6^2+x_7^2)+7.4777(x_6^3+x_7^3) \\
& +0.7854(x_4x_6^2+x_5x_7^2) \\
\text{s.t.} \quad & g_1(\mathbf{x}) = \frac{27}{x_1x_2^2x_3} - 1 \le \mathbf{0}, \ g_2(\mathbf{x}) = \frac{397.5}{x_1x_2^2x_3^2} - 1 \le \mathbf{0}, \\
& g_3(\mathbf{x}) = \frac{1.93x_4^3}{x_2x_3x_6^4} - 1 \le \mathbf{0}, \ g_4(\mathbf{x}) = \frac{1.93x_5^3}{x_2x_3x_7^4} - 1 \le \mathbf{0}, \\
& g_5(\mathbf{x}) = \frac{((\frac{745x_4}{x_2x_3})^2+16.9\times10^6)^{0.5}}{110x_6^3} - 1 \le \mathbf{0}, \\
& g_6(\mathbf{x}) = \frac{((\frac{745x_5}{x_2x_3})^2+157.5\times10^6)^{0.5}}{85x_7^3} - 1 \le \mathbf{0}, \\
& g_7(\mathbf{x}) = \frac{x_2x_3}{40} - 1 \le \mathbf{0}, \ g_8(\mathbf{x}) = \frac{5x_2}{x_1} - 1 \le \mathbf{0}, \\
& g_9(\mathbf{x}) = \frac{x_1}{12x_2} - 1 \le \mathbf{0}, \ g_{10}(\mathbf{x}) = \frac{1.5x_6+1.9}{x_4} - 1 \le \mathbf{0}, \\
& g_{11}(\mathbf{x}) = \frac{1.1x_7+1.9}{x_5} - 1 \le \mathbf{0}
\end{aligned}$$

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 (Kazemi et al. 2011; Liu et al. 2017). The total cost of material, forming, and welding of a cylindrical vessel is minimized. This problem has four design variables and six constraints The optimization problem formulated as following:

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

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 (Kazemi et al. 2011; Liu et al. 2017). The weight subject to constraints on minimum deflection, shear stress, surge frequency, and limits on outside diameter is minimized. This problem has three design variables and four constraints. The optimization problem formulated as following:

$$\begin{aligned}
\min \quad & f(\mathbf{x}) = x_1^2x_2(x_3+2) \\
\text{s.t.} \quad & g_1(\mathbf{x}) = 1 - \frac{x_2^3x_3}{71875x_1^4} \le \mathbf{0}, \\
& g_2(\mathbf{x}) = \frac{x_2(4x_2-x_1)}{12566x_1^3(x_2-x_1)} + \frac{2.46}{12566x_1^2} - 1 \le \mathbf{0}, \\
& g_3(\mathbf{x}) = 1 - \frac{140.54x_1}{x_3x_2^2} \le \mathbf{0}, \ g_4(\mathbf{x}) = \frac{x_1+x_2}{1.5} - 1 \le \mathbf{0}
\end{aligned}$$

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 (Liu et al. 2017; Ray and Liew 2003). The volume subject to stress constraints is minimized. This problem has two design variables and three constraints. The optimization problem formulated as following:

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

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

Table 10 Key characteristics of the optimization test problems with equality constraints

(#)	Label	Source	n	C. type	Variable bounds (D)	Optimum (f^*)
1e	G03	(Liu et al. 2017)	10	NL	$[0, 10]^n$	− 1.0005
2e	G05	(Liu et al. 2017)	4	NL	$[10, 1{,}200]^2 \times [-0.55, 0.55]^2$	5126.4967
3e	G11	(Liu et al. 2017)	2	NL	$[-1, 1]^n$	0.7499
4e	G13	(Liu et al. 2017)	5	NL	$[-2.3, 2.3]^2 \times [-3.2, 3.2]^3$	0.0539
5e	P01	(Birgin et al. 2010)	5	NL	$[-5, 5]^n$	0.0293
6e	P02(a)	(Birgin et al. 2010)	9	NL	$[0, 100] \times [0, 500]^8$	− 400.0000
7e	P02(b)	(Birgin et al. 2010)	9	NL	$[0, 600] \times [0, 500]^8$	− 600.0000
8e	P02(c)	(Birgin et al. 2010)	9	NL	$[0, 100] \times [0, 500]^8$	− 750.0000
9e	P02(d)	(Birgin et al. 2010)	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)	(Birgin et al. 2010)	6	NL	$[0, 1]^4 \times [10^{(-5)}, 16]^2$	0.3888
11e	P05	(Birgin et al. 2010)	3	NL	$[0, 9.422] \times [0, 5.903] \times [0, 267.42]$	201.1600
12e	P09	(Birgin et al. 2010)	6	L	$[10^{(-5)}, 3] \times [10^{(-5)}, 4]^2 \times [0, 2]^2 \times [0, 6]$	− 13.4020
13e	P12	(Birgin et al. 2010)	2	NL	$[0, 2] \times [0, 3]$	−16.7390
14e	P13	(Birgin et al. 2010)	3	NL	$[10^{(-5)}, 34] \times [10^{(-5)}, 17] \times [100, 300]$	189.3500
15e	P14	(Birgin et al. 2010)	4	L	$[10^{(-5)}, 3] \times [10^{(-5)}, 4] \times [0, 2] \times [0, 1]$	− 4.51420
16e	P15	(Birgin et al. 2010)	3	NL	$[10^{(-5)}, 12.5] \times [10^{(-5)}, 37.5] \times [0, 50]$	0.0000
17e	P16	(Birgin et al. 2010)	5	L	$[0, 1.5834] \times [0, 3.625] \times [0, 1] \times [0, 3] \times [0, 4]$	0.7049

Appendix B: Test problems with linear and nonlinear constraints

Table 11 Key characteristics of the constrained global optimization test problems

(#)	Label	Source	n	C. type	Variable bounds (D)	Optimum (f^*)
1	Bunnag 1	(Vaz and Vicente 2009)	4	L	$[0, 3]^n$	0.1117
2	Bunnag 2	(Vaz and Vicente 2009)	4	L	$[0, 4]^n$	− 6.4049
3	Bunnag 3	(Vaz and Vicente 2009)	5	L	$[0, 3] \times [0, 2] \times [0, 4] \times [0, 4] \times [0, 2]$	− 16.3657
4	Bunnag 4	(Vaz and Vicente 2009)	6	L	$[0, 1]^5 \times [0, 20]$	− 213.0470
5	Bunnag 5	(Vaz and Vicente 2009)	6	L	$[0, 2] \times [0, 8] \times [0, 2] \times [0, 1] \times [0, 1] \times [0, 2]$	− 11.0000
6	Bunnag 6	(Vaz and Vicente 2009)	10	L	$[0, 1]^n$	− 268.0146
7	Bunnag 7	(Vaz and Vicente 2009)	10	L	$[0, 1]^n$	− 39.0000
8	G01	(Liu et al. 2017)	13	L	$[0, 10]^9 \times [0, 100]^3 \times [0, 10]$	− 15.0000
9	G02	(Liu et al. 2017)	20	NL	$[0, 10]^n$	− 0.8036
10	G04	(Liu et al. 2017)	5	NL	$[78, 102] \times [33, 45] \times [27, 45]^3$	− 30665.5386
11	G06	(Liu et al. 2017)	2	NL	$[13, 100] \times [0, 100]$	− 6961.8138
12	G07	(Liu et al. 2017)	10	NL	$[-10, 10]^n$	24.3062
13	G08	(Liu et al. 2017)	2	NL	$[0, 10]^n$	− 0.0958
14	G09	(Liu et al. 2017)	7	NL	$[-10, 10]^n$	680.6300
15	G10	(Liu et al. 2017)	8	NL	$[100; 10{,}000] \times [1{,}000; 10{,}000]^2 \times [10; 1{,}000]^5$	7049.2480
16	G12*	(Liu et al. 2017)	3	NL	$[0.2, 10]^n$	−1.0000
17	G16	(Suganthan et al. 2005)	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

Table 11 (continued)

(#)	Label	Source	n	C. type	Variable bounds (D)	Optimum (f^*)
18	G18	(Suganthan et al. 2005)	9	NL	$[0, 10]^n$	− 0.8660
19	G19	(Suganthan et al. 2005)	15	NL	$[0, 10]^n$	32.6555
20	G24	(Suganthan et al. 2005)	2	NL	$[0, 3] \times [0, 4]$	− 5.5080
21	Genocop 9	(Vaz and Vicente 2009)	3	L	$[0, 10]^n$	− 2.4714
22	Genocop 10	(Vaz and Vicente 2009)	4	L	$[0, 3] \times [0, 10] \times [0, 10] \times [0, 1]$	− 4.5280
23	Genocop 11	(Vaz and Vicente 2009)	6	L	$[0, 5] \times [0, 8] \times [0, 5] \times [0, 1] \times [0, 1] \times [0, 2]$	− 11.0000
24	Goldstein & Price	(Na et al. 2017)	2	NL	$[-2, 2]^n$	3.5389
25	Himmelblau	(Cagnina et al. 2008)	5	NL	$[78, 102] \times [33, 45] \times [27, 45]^3$	− 31025.5602
26	Horst 1	(Horst et al. 1995)	2	L	$[0, 3] \times [0, 2]$	− 1.0625
27	Horst 2	(Horst et al. 1995)	2	L	$[0, 2.5] \times [0, 2]$	−6.8995
28	Horst 3	(Horst et al. 1995)	2	L	$[0, 1] \times [0, 1.5]$	− 0.4444
29	Horst 4	(Horst et al. 1995)	3	L	$[0.5, 2] \times [0, 3] \times [0, 2.8]$	−6.0858
30	Horst 5	(Horst et al. 1995)	3	L	$[0, 1.2] \times [0, 1.2] \times [0, 1.7]$	−3.7220
31	Horst 6	(Horst et al. 1995)	3	L	$[0, 6] \times [0, 5.0279] \times [0, 2.6]$	− 32.5784
32	Horst 7	(Horst et al. 1995)	3	L	$[0, 6] \times [0, 3] \times [0, 3]$	− 52.8769
33	hs021	(Vaz and Vicente 2009)	2	L	$[2, 50] \times [-50, 10]$	− 99.9599
34	hs021mod	(Vaz and Vicente 2009)	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	(Vaz and Vicente 2009)	2	L	$[0, 5]^n$	− 1.0000
36	hs035	(Vaz and Vicente 2009)	3	L	$[0, 3]^n$	0.1111
37	hs036	(Vaz and Vicente 2009)	3	L	$[0, 20] \times [0, 11] \times [0, 15]$	− 3300.0000
38	hs037	(Vaz and Vicente 2009)	3	L	$[0, 42]^n$	− 3456.0000
39	hs038	(Vaz and Vicente 2009)	4	L	$[-10, 10]^n$	0.0000
40	hs044	(Vaz and Vicente 2009)	4	L	$[0, 5]^n$	− 15.0000
41	hs076	(Vaz and Vicente 2009)	4	L	$[0, 1] \times [0, 3] \times [0, 1] \times [0, 1]$	− 4.6818
42	s224	(Vaz and Vicente 2009)	2	L	$[0, 6] \times [0, 11]$	− 304.0000
43	s231	(Vaz and Vicente 2009)	2	L	$[-10, 10]^n$	0.0000
44	s232	(Vaz and Vicente 2009)	2	L	$[0, 100]^n$	− 1.0000
45	s250	(Vaz and Vicente 2009)	3	L	$[0, 20] \times [0, 11] \times [0, 42]$	− 3300.0000
46	s251	(Vaz and Vicente 2009)	3	L	$[0, 42]^n$	− 3456.0000
47	T1 ($n = 2$)	(Finkel 2005)	2	NL	$[-4, 4]^n$	− 3.4641
48	T1 ($n = 3$)	(Finkel 2005)	3	NL	$[-4, 4]^n$	− 4.2426
49	T1 ($n = 4$)	(Finkel 2005)	4	NL	$[-4, 4]^n$	− 4.8989
50	T1 ($n = 5$)	(Finkel 2005)	5	NL	$[-4, 4]^n$	−5.4772
51	T1 ($n = 6$)	(Finkel 2005)	6	NL	$[-4, 4]^n$	− 6.0000
52	T1 ($n = 7$)	(Finkel 2005)	7	NL	$[-4, 4]^n$	− 6.4807
53	T1 ($n = 8$)	(Finkel 2005)	8	NL	$[-4, 4]^n$	− 6.9282
54	T1 ($n = 9$)	(Finkel 2005)	9	NL	$[-4, 4]^n$	− 7.3484
55	T1 ($n = 10$)	(Finkel 2005)	10	NL	$[-4, 4]^n$	− 7.7460
56	zecevic2	(Vaz and Vicente 2009)	2	L	$[0, 10]^n$	− 4.1249
57	P03(b)	(Birgin et al. 2010)	2	NL	$[10^{(-5)}, 16]^n$	0.3888
58	P04	(Birgin et al. 2010)	2	NL	$[0, 6] \times [0, 4]$	− 6.6666
59	P06	(Birgin et al. 2010)	2	NL	$[0, 115.8] \times [10^{(-5)}, 30]$	376.2900
60	P07	(Birgin et al. 2010)	2	NL	$[-2, 2]^n$	− 2.8284
61	P08	(Birgin et al. 2010)	2	NL	$[-8, 10] \times [0, 10]$	− 118.7000
62	P10	(Birgin et al. 2010)	2	NL	$[0, 1]^n$	0.7417
63	P11	(Birgin et al. 2010)	2	NL	$[0, 1]^n$	− 0.5000

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

References

Basudhar A, Dribusch C, Lacaze S, Missoum S (2012) Constrained efficient global optimization with support vector machines. Struct Multidiscip Optim 46(2):201–221. https://doi.org/10.1007/s00158-011-0745-5

Biegler LT, Grossmann IE (2004) Retrospective on optimization. Comput Chem Eng 28(8):1169–1192. https://doi.org/10.1016/j.compchemeng.2003.11.003

Birgin EG, Floudas CA, Martínez JM (2010) Global minimization using an augmented lagrangian method with variable lower-level constraints. Math Programming 125(1):139–162. https://doi.org/10.1007/s10107-009-0264-y

Cagnina LC, Esquivel SC, Coello CAC (2008) Solving engineering optimization problems with the simple constrained particle swarm optimizer. Informatica (Ljubljana) 32(3):319–326

Costa MFP, Rocha AMAC, Fernandes EMGP (2017) Filter-based direct method for constrained global optimization. Journal of Global Optimization in Press. https://doi.org/10.1007/s10898-017-0596-8

Dolan ED, Moré JJ (2002) Benchmarking optimization software with performance profiles. Math Program 91(2):201–213. https://doi.org/10.1007/s101070100263

Finkel DE (2005) Global optimization with the DIRECT algorithm. Ph.D. thesis, North Carolina State University

Fletcher R (1987) Practical methods of optimization, 2nd. John and Sons, Chichester. https://doi.org/10.1097/00000539-200101000-00069

Fletcher R, Leyffer S (2002) Nonlinear programming without a penalty function. Math Programming 91(2):239–269. https://doi.org/10.1007/s101070100244

Floudas CA (1999) Deterministic global optimization: theory, methods and applications, nonconvex optimization and its applications, vol 37. Springer, New York. https://doi.org/10.1007/978-1-4757-4949-6

Forrester AIJ, Keane AJ (2009) Recent advances in surrogate-based optimization. Prog Aerosp Sci 45(1):50–79. https://doi.org/10.1016/j.paerosci.2008.11.001

Gablonsky JM (2001) Modifications of the DIRECT algorithm. Ph.D. thesis, North Carolina State University

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

Horst R, Pardalos PM, Thoai NV (1995) Introduction to global optimization. Nonconvex optimization and its application. Kluwer Academic Publishers, Dordrect

Jones DR (2001) The DIRECT global optimization algorithm. In: Floudas CA, Pardalos PM (eds) The encyclopedia of optimization. Kluwer Academic Publishers, Dordrect, pp 431-440

Jones DR, Perttunen CD, Stuckman BE (1993) Lipschitzian optimization without the Lipschitz constant. J Optim Theory Appl 79(1):157–181. https://doi.org/10.1007/BF00941892

Kazemi M, Wang GG, Rahnamayan S, Gupta K (2011) Metamodel-based optimization for problems with expensive objective and constraint functions. J Mech Des 133(1):14,505. https://doi.org/10.1115/1.4003035

Liu H, Xu S, Chen X, Wang X, Ma Q (2017) Constrained global optimization via a direct-type constraint-handling technique and an adaptive metamodeling strategy. Struct Multidiscip Optim 55(1):155–177. https://doi.org/10.1007/s00158-016-1482-6

Liu Q, Cheng W (2014) A modified DIRECT algorithm with bilevel partition. J Glob Optim 60(3):483–499. https://doi.org/10.1007/s10898-013-0119-1

Moré JJ, Wild SM (2009) Benchmarking derivative-free optimization algorithms. SIAM J Optim 20(1):172–191. https://doi.org/10.1137/080724083

Na J, Lim Y, Han C (2017) A modified DIRECT algorithm for hidden constraints in an LNG process optimization. Energy 126(C):488–500

Paulavičius R, Chiter L, Žilinskas J (2018) Global optimization based on bisection of rectangles, function values at diagonals, and a set of Lipschitz constants. J Glob Optim 71(1):5–20. https://doi.org/10.1007/s10898-016-0485-6

Paulavičius R, Sergeyev YD, Kvasov DE, Žilinskas J (2014) Globally-biased DISIMPL algorithm for expensive global optimization. J Glob Optim 59(2-3):545–567. https://doi.org/10.1007/s10898-014-0180-4

Paulavičius R, Žilinskas J (2013) Simplicial Lipschitz optimization without the Lipschitz constant. J Glob Optim 59(1):23–40. https://doi.org/10.1007/s10898-013-0089-3

Paulavičius R, Žilinskas J (2014) Simplicial global optimization. Springer Briefs in optimization. Springer, New York. https://doi.org/10.1007/978-1-4614-9093-7

Paulavičius R, Žilinskas J (2016) Advantages of simplicial partitioning for Lipschitz optimization problems with linear constraints. Optim Lett 10(2):237–246. https://doi.org/10.1007/s11590-014-0772-4

Pillo GD, Liuzzi G, Lucidi S, Piccialli V, Rinaldi F (2016) A direct-type approach for derivative-free constrained global optimization. Comput Optim Appl 65(2):361–397. https://doi.org/10.1007/s10589-016-9876-3

Pillo GD, Lucidi S, Rinaldi F (2010) An approach to constrained global optimization based on exact penalty functions. J Optim Theory Appl 54(2):251–260. https://doi.org/10.1007/s10898-010-9582-0

Pintér JD (1996) Global optimization in action: continuous and Lipschitz optimization: algorithms, implementations and applications, nonconvex optimization and its applications, vol 6. Springer, New York. https://doi.org/10.1007/978-1-4757-2502-5

Ray T, Liew KM (2003) Society and civilization: an optimization algorithm based on the simulation of social behavior. IEEE Trans Evol Comput 7(4):386–396. https://doi.org/10.1109/TEVC.2003.814902

Regis RG (2011) Stochastic radial basis function algorithms for large-scale optimization involving expensive black-box objective and constraint functions. Comput Oper Res 38(5):837–853. https://doi.org/10.1016/j.cor.2010.09.013

Regis RG (2014) Constrained optimization by radial basis function interpolation for high-dimensional expensive black-box problems with infeasible initial points. Eng Optim 46(2):218–243. https://doi.org/10.1080/0305215X.2013.765000

Sergeyev YD, Kvasov DE (2006) Global search based on diagonal partitions and a set of Lipschitz constants. SIAM J Optim 16(3):910–937. https://doi.org/10.1137/040621132

Shan S, Wang GG (2010) Metamodeling for high dimensional simulation-based design problems. J Mech Des 132(5):051,009. https://doi.org/10.1115/1.4001597

Shan S, Wang GG (2010) Survey of modeling and optimization strategies to solve high-dimensional design problems with computationally-expensive black-box functions. Struct Multidiscip Optim 41(2):219–241. https://doi.org/10.1007/s00158-009-0420-2

Stripinis L, Paulavičius R, Žilinskas J (2018) Improved scheme for selection of potentially optimal hyper-rectangles in direct. Optim Lett 12(7):1699–1712. https://doi.org/10.1007/s11590-017-1228-4

Stripinis L, Paulavičius R (2018) DIRECTLib – a library of global optimization problems for DIRECT-type methods, v1.1. https://doi.org/10.5281/zenodo.1403547
Suganthan PN, Hansen N, Liang JJ, Deb K, Chen YP, Auger A, Tiwari S (2005) Problem definitions and evaluation criteria for the cec 2006 special session on constrained real-parameter optimization. KanGAL, pp 251–256
Vaz A, Vicente L (2009) Pswarm: a hybrid solver for linearly constrained global derivative-free optimization. Optim Methods Softw 24(4–5):669–685. https://doi.org/10.1080/10556780902909948