Journal of Computational and Applied Mathematics 236 (2012) 4042–4054

Contents lists available at SciVerse ScienceDirect

Journal of Computational and Applied Mathematics

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

Lipschitz gradients for global optimization in a one-point-based partitioning scheme☆

Dmitri E. Kvasov, Yaroslav D. Sergeyev *

DEIS – University of Calabria, Via P. Bucci 42C, 87036 – Rende (CS), Italy
Software Department, N.I. Lobachevsky State University, Nizhni Novgorod, Russia

ARTICLE INFO

Article history:
Received 31 December 2011
Received in revised form 10 February 2012

MSC:
65K05
90C26
90C56

Keywords:
Global optimization
Lipschitz gradients
Set of Lipschitz constants
Geometric algorithms

ABSTRACT

A global optimization problem is studied where the objective function $f(x)$ is a multidimensional black-box function and its gradient $f'(x)$ satisfies the Lipschitz condition over a hyperinterval with an unknown Lipschitz constant K. Different methods for solving this problem by using an a priori given estimate of K, its adaptive estimates, and adaptive estimates of local Lipschitz constants are known in the literature. Recently, the authors have proposed a one-dimensional algorithm working with multiple estimates of the Lipschitz constant for $f'(x)$ (the existence of such an algorithm was a challenge for 15 years). In this paper, a new multidimensional geometric method evolving the ideas of this one-dimensional scheme and using an efficient one-point-based partitioning strategy is proposed. Numerical experiments executed on 800 multidimensional test functions demonstrate quite a promising performance in comparison with popular DIRECT-based methods.

1. Introduction

Global optimization is an important part of numerical analysis (see, e.g., [1–4]). It considers problems derived from complex industrial applications where the objective function $f(x)$ to be minimized is defined over a hyperinterval $D \subset R^N$, $N \geq 1$, and can be black-box, multiextremal, and requiring high computational resources for its evaluation (see, e.g., [1,2,4–10]). Solving efficiently this type of problems is a great challenge, since they present a high number of local minimizers (only a few of which can be global ones), often with extremely different values, and do not present a simple mathematical description of the global optima.

One of the natural and powerful (from both the theoretical and the applied points of view) assumptions on these problems is that the objective function has bounded slopes, i.e.,

$$|f(x') - f(x'')| \leq L\|x' - x''\|, \quad x', x'' \in D, 0 < L < \infty, \tag{1}$$

where $\|\cdot\|$ denotes, usually, the Euclidean norm (other norms can be also used, see, e.g., [11,12]) and L is the (unknown) Lipschitz constant. In this case, Lipschitz global optimization methods can be applied (see, e.g., [2,4,9,10,13–15] and the references given therein). They can be distinguished, for example, by the way in which information about the Lipschitz constant is obtained and by the strategy of exploration of the admissible region.

☆ This work was supported by the grants 1960.2012.9 and MK-3473.2010.1 awarded by the President of the Russian Federation for supporting the leading research groups and young researchers, respectively, as well as by the grant 11-01-00682-a awarded by the Russian Foundation for Fundamental Research. The authors also thank the Italian Inter-University Consortium for the Application of Super-Computing for Universities and Research (CASPUR), project "Advanced techniques for global optimization: Numerical methods and applications" in the framework of the "HPC Grant 2011 on GPU cluster".

* Corresponding author at: DEIS – University of Calabria, Via P. Bucci 42C, 87036 – Rende (CS), Italy. Fax: +39 0984 494855.
E-mail addresses: kvadim@si.deis.unical.it (D.E. Kvasov), yaro@si.deis.unical.it (Y.D. Sergeyev).

0377-0427/$ – see front matter
doi:10.1016/j.cam.2012.02.020

In the literature, there exist at least four approaches to specify the Lipschitz constant L from (1): (i) it can be given a priori (see, e.g., [13,14,16]); (ii) its global estimate over the whole domain can be used (see, e.g., [2,9,15,17]); (iii) local Lipschitz constants can be estimated (see, e.g., [2,10,18,19]); (iv) several estimates of L can be chosen from a set of possible values (see, e.g., [10,20–25]). In their work global optimization methods using multiple estimates of the Lipschitz constants have proved to be particularly attractive for studying applied problems (see, e.g., [22,26–29], other references can be found, e.g., in [25,30]).

In exploring the multidimensional search domain, various adaptive partitioning strategies can be applied. For example, one-point-based algorithms subsequently subdivide the search region in smaller ones and evaluate the objective function at one point within each subregion (see, e.g., [13,21,23,31,32]). Partitions of the search domain into hyperintervals, based on evaluating the objective function at the two vertices corresponding to the main diagonal of hyperintervals called *diagonal partitioning strategies*, can also be successfully used (see, e.g., [9,10,17,18,25,33]). More complex partitions, based on simplices, auxiliary functions of various nature, and so on, have also been proposed (see, e.g., [4,15,34,35]; many other references can be found in [36]).

The choice of the regions to be partitioned is based on an information about the objective function obtained during the search. It can be either of the probabilistic type (e.g., Bayesian approach applying the theory of random functions to a mathematical representation of available (certain or uncertain) a priori information on the objective function behavior, see, e.g., [2,4,8,37,38]), or of the deterministic one (e.g., geometric approach making a use of different auxiliary functions to estimate the behavior of $f(x)$ over the search region, see, e.g., [2,4,9,10,14,15,19,39,40]).

In this paper, a particular class of the Lipschitz global optimization problems is considered, namely, the class of problems with differentiable objective functions having the Lipschitz gradients $f'(x)$, i.e.,

$$f^* = f(x^*) = \min_{x \in D} f(x), \tag{2}$$

$$\|f'(x') - f'(x'')\| \leq K\|x' - x''\|, \quad x', x'' \in D, 0 < K < \infty, \tag{3}$$

where

$$D = [a, b] = \{x \in R^N : a(j) \leq x(j) \leq b(j)\}. \tag{4}$$

It is supposed in this formulation that the objective function $f(x)$ can be black-box, multiextremal, its gradient $f'(x) = \left(\frac{\partial f(x)}{\partial x(1)}, \frac{\partial f(x)}{\partial x(2)}, \ldots, \frac{\partial f(x)}{\partial x(N)}\right)^T$ (which could be itself a costly multiextremal black-box vector-function) can be calculated during the search, and $f'(x)$ is Lipschitz-continuous with some fixed, but unknown, constant $K, 0 < K < \infty$, over D. These problems are often encountered in engineering applications (see, e.g., [2,9,10]), particularly, in electrical engineering optimization problems (see, e.g., [2,10,41]).

In the literature, several methods for solving this problem have been proposed. They can be also distinguished, for instance, with respect to the way the Lipschitz constant K is estimated in their work. There exist algorithms using an a priori given estimate of K (see, e.g., [39,40,42]), its adaptive estimates (see, e.g., [10,33,40]), and adaptive estimates of local Lipschitz constants (see, e.g., [10,40]). Algorithms working with a number of Lipschitz constants for $f'(x)$ chosen from a set of possible values varying from zero to infinity were not known till 2009 when such an algorithm for solving the one-dimensional problem (2)–(4) has been proposed in [30]. Its extension to the multidimensional case is not a trivial task in contrast to the DIRECT method (see [23]) proposed in 1993 for solving problems with the Lipschitz objective function.

The present paper solves this more than 15-year open problem of constructing multidimensional global optimization methods working with multiple estimates of the Lipschitz constants for $f'(x)$. A new multidimensional geometric method for finding solutions to the problem (2)–(4) is introduced and studied here. It uses a new one-point-based partitioning strategy (see [10,32]) and works with a number of estimates of the Lipschitz constant K for $f'(x)$. Such multiple (from zero to infinity) estimates of K from (3) are used to calculate the lower bounds of the objective function over the hyperintervals of a current partition of the search domain and to produce new trial points (i.e., points at which both the objective function $f(x)$ and its gradient $f'(x)$ are evaluated). In the framework of geometric algorithms, this kind of estimating the Lipschitz constant can be interpreted as examination of all admissible minorant functions during the current iteration of the algorithm without constructing a specific one. A particular attention in the new algorithm is given to the improvement of the current minimal function value (the so-called *record value*) in order to provide a faster convergence to a global minimizer. As demonstrated by extensive numerical experiments executed on 800 test functions from the differentiable GKLS test classes (see [43]), the usage of gradients allows one to obtain, as expected, an acceleration in comparison with the DIRECT-based methods.

The paper is organized as follows. In Section 2, a theoretical background of the new algorithm is presented. Section 3 is dedicated to the description of the algorithm and to its convergence analysis. Finally, Section 4 contains results of numerical experiments executed on 800 test functions.

2. Theoretical background

In this section, the main theoretical results, necessary for introducing the new algorithm, are obtained. First, a new partitioning strategy developed in the framework of the one-point-based partition approach is described. The second part presents a technique for estimating the lower bounds of the objective function over hyperintervals. The third part is

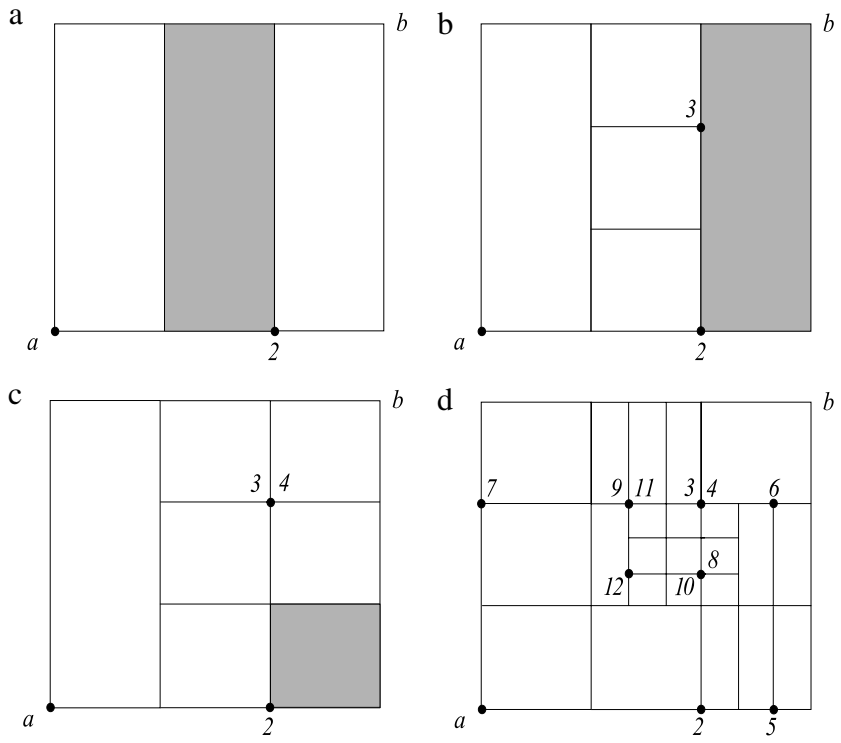

Fig. 1. An example of subdivisions by a new one-point-based partitioning strategy.

dedicated to the introduction of a procedure for determining nondominated hyperintervals, i.e., hyperintervals having the smallest lower bound for some particular estimate of the Lipschitz constant for $f'(x)$. They are candidates for partitioning at each iteration of the new method.

2.1. One-point-based partitioning strategy

In this section, a new efficient one-point-based partitioning scheme proposed in [32] (see also [10]) is considered which is based on a diagonal partitioning strategy from [10,44]. In this scheme, the function $f(x)$ and its gradient $f'(x)$ are evaluated only at one vertex (either a_i or b_i) of the main diagonal of each hyperinterval $D_i = [a_i, b_i]$ of the current partition independently of the problem dimension (recall that performing each trial is a time-consuming operation).

Let us start the description of this scheme with a two-dimensional example shown in Fig. 1. In this Figure, partitions of the admissible region D produced by the algorithm at several initial iterations are presented starting from the first trial at the point a (it is supposed here that a single iteration consists of the subdivision of only one hyperinterval). Black dots represent the trial points and the numbers around these dots indicate iterations at which these trial points have been generated. The terms 'interval' and 'subinterval' will be used to denote two-dimensional rectangular domains.

In Fig. 1a, the situation after the first two iterations is presented. Particularly, at the second iteration, the interval D is partitioned into three subintervals of equal area (equal volume in a general case). This subdivision is performed by two lines (hyperplanes) orthogonal to the longest edge of D (see Fig. 1a). The trial (evaluation of the objective function and, as we propose in this paper, of its gradient) is performed only at the point denoted by number 2.

Let us suppose that the interval shown in light gray in Fig. 1a is chosen for the further partitioning. Thus, at the third iteration, three smaller subintervals are generated (see Fig. 1b). As one can see from Fig. 1c, the trial point of the fourth iteration coincides with the point 3 at which the trial has already been executed. Therefore, there is no need to perform a new (costly) evaluation of $f(x)$ and $f'(x)$ at this point, since the values obtained at the previous iteration can be used. These values can be stored in a specially designed vertex database and is simply retrieved on demand without re-evaluations of the functions. For example, Fig. 1d illustrates the situation after 12 iterations. It can be seen from this figure that 23 intervals have been generated by only 9 trial points.

Now we can describe the general scheme of a hyperinterval partitioning by assuming (without loss of generality) that the search hyperinterval D in (4) is an N-dimensional hypercube and the first trial is performed at the vertex a (the scheme starting from the vertex b is obtained analogously). Let a hyperinterval $D_t = [a_t, b_t]$ of a current partition $\{D^k\}$ of $D = [a, b]$ be chosen for partitioning at an iteration $k \geq 1$ of the algorithm. The operation of partitioning the selected hyperinterval D_t is performed as follows.

Step 1. Determine points u and v by the following formulas

$$u = \left(a(1), \ldots, a(i-1), a(i) + \frac{2}{3}(b(i) - a(i)), a(i+1), \ldots, a(N)\right), \tag{5}$$

$$v = \left(b(1), \ldots, b(i-1), b(i) + \frac{2}{3}(a(i) - b(i)), b(i+1), \ldots, b(N)\right), \tag{6}$$

where $a(j) = a_t(j)$, $b(j) = b_t(j)$, $1 \le j \le N$, and i is given by the equation

$$i = \arg\min \max_{1 \le j \le N} |b(j) - a(j)|. \tag{7}$$

Get (evaluate or read from the vertex database) the values of the objective function $f(x)$ and its gradient $f'(x)$ only at the point u.

Step 2. Divide the hyperinterval D_t into three hyperintervals of equal volume by two parallel hyperplanes that are perpendicular to the longest edge i of D_t and pass through the points u and v.

The hyperinterval D_t is so substituted by three new hyperintervals with indices $t' = t$, $m+1$, and $m+2$ (where $m = m(k)$ is the number of hyperintervals at the beginning of the iteration k) determined by the vertices of their main diagonals

$$a_{t'} = a_{m+2} = u, \qquad b_{t'} = b_{m+1} = v, \tag{8}$$

$$a_{m+1} = a_t, \qquad b_{m+1} = v, \tag{9}$$

$$a_{m+2} = u, \qquad b_{m+2} = b_t. \tag{10}$$

Augment the current number of hyperintervals m by 2.

From the partitioning scheme described above it can be observed that, contrary to many traditional partitioning strategies (see, e.g., [9,18,23,31,33]), the condition

$$a_i(j) < b_i(j) \quad \forall j : j = 1, \ldots, N,$$

does not have to be satisfied for all hyperintervals $D_i \subset D$, and their main diagonals determined by the vertices a_i and b_i can be oriented in different ways. However, as theoretically shown in [10,44], the hyperintervals orientations are not arbitrary and a special linking of hyperintervals generated at different iterations can be established with some efforts.

This smart linking will allow us to store information about vertices and the corresponding values of $f(x)$ and $f'(x)$ in a special database, thereby avoiding redundant functions evaluations. The objective function and its gradient will be calculated at a vertex only once, stored in the database, and read when required. The new partitioning strategy generates trial points in such a regular way that one vertex where the functions are evaluated can belong to several (up to 2^N) hyperintervals (see, for example, a trial point at the 8-th iteration in Fig. 1d). Therefore, the time-consuming operation of the functions evaluations is replaced by a significantly faster operation of reading (up to 2^N times) the functions values from the database. In this way, the new partitioning strategy considerably speeds up the search, especially when problems of high dimensions are considered (see [17,32,44]).

Note also that the possibility to choose the sequence of trial points among either the points a_i or the points b_i (or among other $2^N - 2$ vertices) of hyperintervals D_i (see Step 1 of the scheme) offers an important tool for accelerating the global search when some additional information about the objective function is known (we will see an example of this situation in Section 4). Note the center-sampling partitioning strategies (see, e.g., [21,23,31]) do not have this property.

2.2. Lower bounding

Let us consider an iteration $k \ge 1$ of the new algorithm and a current partition $\{D^k\}$ of the search hyperinterval $D = [a, b]$ into hyperintervals $D_i = [a_i, b_i]$, $1 \le i \le m(k)$; over these hyperintervals the values of both the function and its gradient are obtained (evaluated or read from the vertex database) at trial points $x^{j(k)} = a_i, j(k) \ge 1$. In order to choose some hyperintervals for the further partition, the goodness (expressed by the so-called *characteristic*, see, e.g., [2,9,10,36]) of the hyperintervals with respect to the global search is estimated by the algorithm. Better is the characteristic of a hyperinterval (in some predetermined sense), higher is the possibility to find the global minimizer within this hyperinterval. This hyperinterval is, therefore, a good candidate for a subdivision at the next iteration of the algorithm.

An estimate of the lower bound of $f(x)$ over a hyperinterval is one of the possible characteristics of this hyperinterval. The following result holds.

Theorem 1. *Let $\tilde{K}$ be an estimate of the Lipschitz constant K for $f'(x)$ from (3), $\tilde{K} \ge K$ and $D_i = [a_i, b_i]$ be a hyperinterval of a current partition $\{D^k\}$ with a trial point a_i. Then, a value $R_i(\tilde{K})$ of the characteristic of D_i can be found such that it is the lower bound of $f(x)$ over D_i, i.e., $R_i(\tilde{K}) \le f(x)$, $x \in D_i$.*

Proof. Let us prove the theorem in a constructive way. It is known (see, e.g., [12,31,45]) that for a differentiable function $f(x)$ over a hyperinterval $D_i = [a_i, b_i]$ the following inequality is satisfied:

$$f(x) \ge Q(x, \tilde{K}), \quad x \in D_i, \tag{11}$$

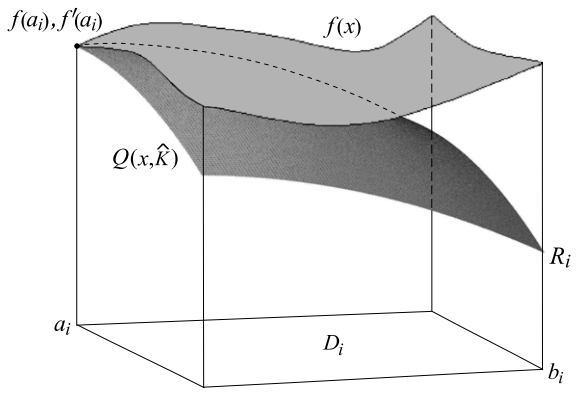

Fig. 2. A quadratic minorant function $Q(x, \tilde{K})$ for $f(x)$ over a hyperinterval $D_i = [a_i, b_i]$.

where the quadratic minorant function $Q(x, \tilde{K})$ is defined over D_i as

$$Q(x, \tilde{K}) = f(a_i) + \langle f'(a_i), (x - a_i)\rangle - 0.5\tilde{K}\|x - a_i\|^2, \quad x \in D_i. \tag{12}$$

Here $\langle\cdot, \cdot\rangle$ is the scalar product, $\|\cdot\|$ is the Euclidean norm in R^N, and

$$g(x) = f(a_i) + \langle f'(a_i), (x - a_i)\rangle$$

is the linear approximation of $f(x)$ over D_i.

From inequality (11) the following estimates can be obtained:

$$\begin{aligned} f(x) &\geq f(a_i) + \langle f'(a_i), (x - a_i)\rangle - 0.5\tilde{K}\|b_i - a_i\|^2 \\ &\geq F_i - 0.5\tilde{K}\|b_i - a_i\|^2, \quad x \in D_i, \end{aligned}$$

where F_i is the minimum value of the linear approximation $g(x)$ over D_i, i.e.,

$$F_i = f(a_i) + \min_{x \in D_i} \langle f'(a_i), (x - a_i)\rangle. \tag{13}$$

Since the function $g(x)$ is linear, its minimum (13) is obtained in the vertex z_i of the hyperinterval $D_i = [a_i, b_i]$ which coordinates $z_i(j), j = 1, \ldots, N$, can be calculated as follows:

$$z_i(j) = \begin{cases} a_i(j), & \text{if either } b_i(j) > a_i(j) \text{ and } \dfrac{\partial f(a_i)}{\partial x(j)} \geq 0, \text{ or } b_i(j) < a_i(j) \text{ and } \dfrac{\partial f(a_i)}{\partial x(j)} < 0; \\ b_i(j), & \text{if either } b_i(j) > a_i(j) \text{ and } \dfrac{\partial f(a_i)}{\partial x(j)} < 0, \text{ or } b_i(j) < a_i(j) \text{ and } \dfrac{\partial f(a_i)}{\partial x(j)} \geq 0. \end{cases} \tag{14}$$

The corresponding value F_i from (13) is therefore equal to

$$F_i = f(a_i) + \langle f'(a_i), (z_i - a_i)\rangle. \tag{15}$$

It is clear now that the value

$$R_i = R_i(\tilde{K}) = F_i - 0.5\tilde{K}\|b_i - a_i\|^2 \tag{16}$$

satisfies the inequality

$$R_i \leq f(x), \quad x \in D_i,$$

and, therefore, it can be taken as the characteristic value of D_i that estimates the lower bound of $f(x)$ over D_i. The theorem has been proved. □

Note that analogous results can be obtained in the case of hyperintervals D_i with trial points b_i rather than a_i.

In Fig. 2, a quadratic minorant function $Q(x, \tilde{K})$ from (12) is illustrated for $f(x)$ over a hyperinterval D_i. Here, the characteristic value R_i coincides with the minimum value of $Q(x, \tilde{K})$ obtained at the point b_i of the main diagonal of D_i. In general, as it can be seen from (12), the value R_i is smaller than or equal to the minimum value of $Q(x, \tilde{K})$ over D_i.

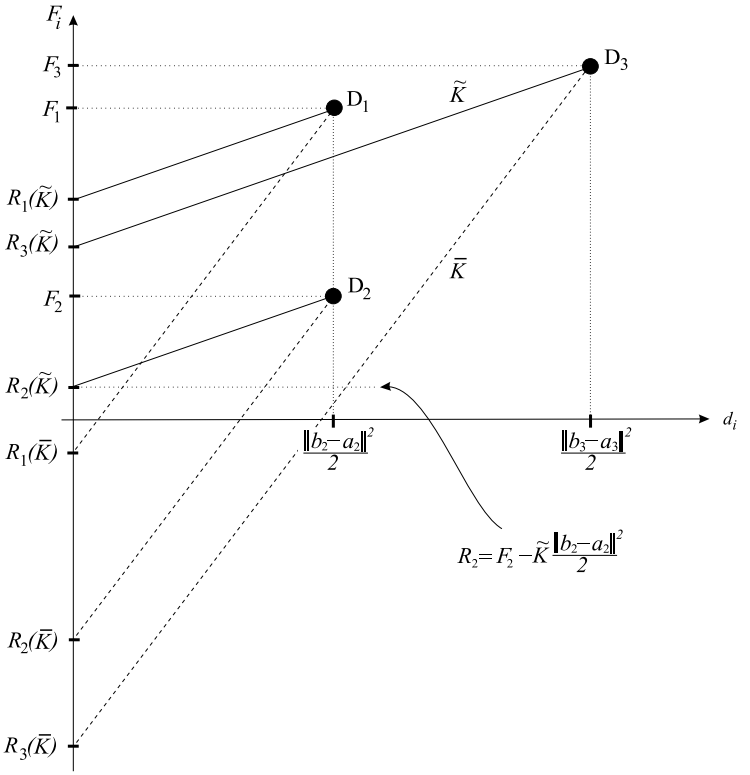

Fig. 3. Graphical representation of hyperintervals.

2.3. Nondominated hyperintervals and their graphical representation

By using the obtained characteristics of hyperintervals, the relation of domination can be established between every two hyperintervals of a current partition $\{D^k\}$ of D and a set of nondominated hyperintervals can be identified for a possible subdivision at the current iteration of the new algorithm (see [25,30]).

Definition 1. Given an estimate $\tilde{K} > 0$ of the Lipschitz constant K from (3), a hyperinterval $D_i = [a_i, b_i]$ *dominates* a hyperinterval $D_j = [a_j, b_j]$ with respect to $\tilde{K}$ if

$$R_i(\tilde{K}) < R_j(\tilde{K}).$$

Definition 2. A hyperinterval $D_t = [a_t, b_t]$ is said to be *nondominated with respect to* $\tilde{K} > 0$ if for the chosen value $\tilde{K}$ there is no other hyperinterval in $\{D^k\}$ which dominates D_t.

Let us now generalize the approach proposed by the authors in [30] for the one-dimensional prototype and show that both a multi-dimensional interval $D_i = [a_i, b_i]$ of a current partition $\{D^k\}$ and the respective characteristic R_i using the gradient can be represented in a two-dimensional diagram similar to those proposed in [23,25] for derivative free methods. Difficulties in the construction of such a diagram were among the main reasons that prevented people to propose methods using several estimates of K in their work.

So, we take for the dot, corresponding to D_i, the vertical coordinate F_i from (13) to (15) and the horizontal coordinate d_i equal to half of the squared length of the main diagonal of D_i, i.e.,

$$d_i = 0.5\|b_i - a_i\|^2.$$

For example, in Fig. 3, a partition of the search domain D consisting of three hyperintervals is represented by the dots D_1, D_2, and D_3. Let us suppose that the Lipschitz constant K for the gradient $f'(x)$ is estimated by $\tilde{K}$, $\tilde{K} \geq K$. The characteristic R_i of a hyperinterval D_i, $i = 1, 2, 3$, can be graphically obtained as the vertical coordinate of the intersection point of the line passed through the point D_i with the slope $\tilde{K}$ and the vertical coordinate axis (see Fig. 3). It is easy to see, that with respect to the estimate $\tilde{K}$ the hyperinterval D_2 dominates both hyperintervals D_1 and D_3 and the hyperinterval D_3 dominates D_1.

If a higher estimate $\bar{K} > \tilde{K}$ of the Lipschitz constant K is considered (see Fig. 3), the hyperinterval D_2 still dominates D_1 with respect to $\bar{K}$, because $R_2(\bar{K}) < R_1(\bar{K})$. But D_2 in its turn is dominated by the hyperinterval D_3 with respect to $\bar{K}$, because $R_2(\bar{K}) > R_3(\bar{K})$ (see Fig. 3).

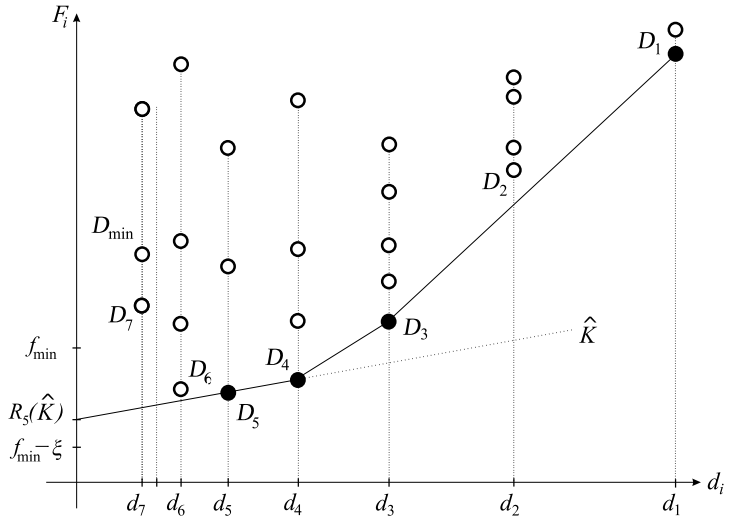

Fig. 4. The two-dimensional diagram representing dominated (white dots) and nondominated (black dots) hyperintervals of a current partition of the search domain.

Since the exact Lipschitz constant K for $f'(x)$ (or its valid overestimate) is unknown in the stated problem, the following definition can be useful.

Definition 3. A hyperinterval $D_t \in \{D^k\}$ is called *nondominated* if there exists an estimate $0 < \tilde{K} < \infty$ of the Lipschitz constant K such that D_t is nondominated with respect to $\tilde{K}$.

This means that nondominated hyperintervals are those with the smallest characteristics (16) for some particular estimate of the Lipschitz constant for the gradient $f'(x)$. For example, in Fig. 3 the hyperintervals D_2 and D_3 are nondominated.

It can be demonstrated following the reasoning used in [25,30] that nondominated hyperintervals (in the sense of Definition 3) are located on the lower-right convex hull of the set of dots representing the hyperintervals of the current partition of D and can be efficiently found by applying algorithm for identifying the convex hull of the dots (see, e.g., [10,22,23]). In Fig. 4, the hyperintervals represented by the dots D_1 (the largest hyperinterval), D_3, D_4, and D_5 are nondominated hyperintervals.

It has been shown in [25] that the hyperintervals of a current partition of D form several groups characterized by the length of their main diagonals. The hyperintervals from a group are represented graphically by dots with the same horizontal coordinate. For example, in Fig. 4 there are seven different groups of hyperintervals with the horizontal coordinates equal to $d_i, i = 1, \ldots, 7$, and one empty group (with the horizontal coordinate between d_7 and d_6 in Fig. 4). Empty groups correspond to hyperintervals which are not present in the current partition but can be generated (or were generated) at the successive (previous) iterations of the method.

As demonstrated in [25,44], a correspondence between the length of the main diagonal of a hyperinterval D_i and a nonnegative integer number can be established, which indicates the number of subdivisions of the initial domain D necessary to obtain the hyperinterval D_i. At each iteration $k \geq 1$ this number can be considered as an index $s(k)$ of a group of equal hyperintervals where

$$0 \leq q_\infty(k) \leq s(k) \leq q_0(k) < +\infty \tag{17}$$

and $q_\infty(k)$ and $q_0(k)$ are indices corresponding to the groups of the largest and smallest hyperintervals of the current partition of D, respectively (for example, in Fig. 4, $q_0(k) = q_\infty(k) + 7$). During partitioning, diagonals of hyperintervals become smaller, while the corresponding group indices grow up consecutively starting from $q_\infty(1) = 0$ (see [25] for details).

Once a nondominated hyperinterval $D_t = [a_t, b_t]$ is determined (with respect to some estimate $\tilde{K}$ of the Lipschitz constant K), it can be subdivided at the next iteration of the algorithm if the following condition is satisfied:

$$R_t(\tilde{K}) \leq f_{\min}(k) - \xi, \tag{18}$$

where R_t is calculated by (16), $f_{\min}(k)$ is the record value, i.e., the current minimal function value (attained at the record point $x_{\min}(k)$), and ξ is the parameter of the algorithm, $\xi \geq 0$ (it can be set in different ways, see Section 4). Notice that both the record value and the record point can be changed after performing some better trial during partitioning, but the record value remains always greater than or equal to the vertical coordinate of the lowest dot (dot D_5 in Fig. 4).

Condition (18) prevents the algorithm from subdividing already well-explored small hyperintervals. For example, among nondominated hyperintervals in Fig. 4 (black dots), the hyperinterval D_5 does not satisfy this condition and therefore is excluded from being partitioned at the next iteration of the method.

It should be mentioned in this occasion that, together with nondominated hyperintervals, a hyperinterval $D_{\min}(k) = [a_{\min}, b_{\min}]$ containing the record point (called hereafter the record hyperinterval) is also considered for a possible partition during the work of the algorithm as it will be explained in the next Section. Among different hyperintervals the record point $x_{\min}(k)$ can belong to (up to 2^N), the record hyperinterval is that with the smallest characteristic and can be changed during subdivisions. In Fig. 4, the record hyperinterval is represented by the dot $D_{\min}$ (note that this dot can be not the lowest one, as in Fig. 4). Hereafter, the index of the group the hyperinterval $D_{\min}(k)$ belongs to will be indicated as $p(k)$ (during the work of the algorithm the satisfaction of inequalities (17) is ensured for this index which can be eventually updated together with $q_0(k)$ and $q_\infty(k)$; see [25] for details). In Fig. 4, $p(k) = q_0(k)$ and, therefore, the hyperinterval $D_{\min}(k)$ is among the smallest hyperintervals of the illustrated partition of D.

3. New algorithm

In this Section, the new algorithm for solving problem (2)–(4) is described. First, the new method is presented and its computational scheme is given, then its convergence properties are analyzed.

The new algorithm consists of the following explicitly defined phases: (1) an exploration phase, at which an examination of large hyperintervals (possibly located far away from the record point) is performed in order to capture new subregions with better function values; (2) a record improvement phase, at which the algorithm tries to better inspect the subregion around the record point. Several subdivisions of different hyperintervals can be performed at a single iteration of the new method (this more general notion of an iteration with respect to that of Section 2.1 is often used in the Lipschitz global optimization algorithms with multiple estimates of Lipschitz constants, see, e.g., [21,23,25,30]).

The exploration phase consists of several iterations (namely, $N + 1$ where N is the problem dimension), each serves for determining nondominated hyperintervals and partitioning them. Since each subdivision of a hyperinterval by the scheme (5)–(10) is performed perpendicularly to only one side of the hyperinterval (to the longest side from (7)), the number of iterations within a phase of the algorithm should be correlated with the hyperintervals dimension.

This phase is interrupted after finishing an iteration if an improvement on at least 1% of the minimal function value is reached, i.e., if

$$f_{\min}(k) \le f_{\min}^{\text{prec}} - 0.01|f_{\min}^{\text{prec}}|, \tag{19}$$

where $f_{\min}^{\text{prec}}$ is the record value memorized at the start of the exploration phase.

Condition (19) is verified after each iteration of the exploration phase and is used to switch the algorithm to the record improvement phase. This local phase is also launched when the exploration phase finishes without having improved the record value, but only if the record hyperinterval $D_{\min}(k)$ is not the smallest one within the current partition of hyperintervals (for example, in Fig. 4, the record hyperinterval is among the smallest hyperintervals). Otherwise, the algorithm re-initiates another global exploration phase without forcing the local one.

The record improvement phase reflects the already well-established fact in global optimization affirming the benefits of the record improvement during the global search (see, e.g., the references given in [10,14,30,38]). At a single iteration, it performs several subdivisions (namely, N) of the record hyperinterval trying to improve the record value. During this process a new record value can appear. In this case, a new record hyperinterval can be considered for remaining subdivisions.

The record hyperinterval subdivisions are performed by means of the one-point-based strategy described in Section 2.1. Of course, other possible local improvement techniques can be used for this scope (see, e.g., [12,45,46]) but in this case the resulting trial points cannot be managed within the vertex database mentioned in Section 2.1.

It is important that the available gradient information allows us to terminate automatically the record improvement phase. In fact, the record hyperinterval is not further subdivided when the gradient projection on the directions parallel to the record hyperinterval sides becomes non-negative, i.e., when the following condition is satisfied:

$$\frac{\partial f(a_{\min})}{\partial x(j)}(b_{\min}(j) - a_{\min}(j)) \ge 0 \quad \forall j : j = 1, \ldots, N. \tag{20}$$

Either in this case or when the prefixed number N of subdivisions are normally performed (without meeting conditions (20)), the algorithm is switched again to the global exploration phase and continues its work.

The algorithm stops when the number of generated trial points reaches the maximal allowed number $P_{\max}$. The satisfaction of this termination criterion is verified after every subdivision of a hyperinterval. The current record value $f_{\min}$ and the current record point $x_{\min}$ can be taken as approximations of the global minimum value f^* and the global minimizer x^* from (2), respectively.

A formal description of the new algorithm follows below (we assume without loss of generality that the admissible region $D = [a, b]$ in (4) is an N-dimensional hypercube).

Step 0 (*Initialization*). Set the iteration counter $k := 1$. Let the first evaluation of $f(x)$ and $f'(x)$ be performed at the vertex a of the initial hyperinterval $D = [a, b]$, i.e., $x^1 := a$. Set the current partition of the search interval as $D^1 := \{[a_1, b_1]\}$, where $a_1 = a$, $b_1 = b$, and the current number of hyperintervals $m(1) := 1$. Set $f_{\min}(1) := f(x^1)$, $x_{\min}(1) := a$, and $D_{\min}(1) := [a_1, b_1]$. Set group indices $q_\infty(1) := q_0(1) := p(1) := 0$.

Suppose now that $k \geq 1$ iterations of the algorithm have already been executed. The next iterations of the algorithm consist of the following steps.

Step 1 (*Exploration Phase*). Memorize the current record $f_{\min}^{\text{prec}} := f_{\min}(k)$, set the counter of iterations during the exploration phase $k_g := 1$ and perform the following steps:

Step 1.1. Identify the set of nondominated hyperintervals considering only groups of large hyperintervals (namely, those with the current indices from $q_\infty(k)$ up to $\lceil (q_\infty(k) + p(k))/2 \rceil$). Subdivide those nondominated hyperintervals which satisfy inequality (18) and produce new trial points (or read the existing ones from the vertex database) according to Section 2.1. Set $k := k + 1$ and update hyperintervals indices if necessary (see [25] for details).

Step 1.2. If condition (19) is satisfied, then go to Step 2 and execute the record improvement phase. Otherwise, go to Step 1.3.

Step 1.3. Increase the counter $k_g := k_g + 1$: check whether $k_g \leq N$. If this is the case, then go to Step 1.1 (continue the exploration of large hyperintervals). Otherwise, go to Step 1.4 (perform the final iteration of the exploration phase by considering more hyperintervals groups).

Step 1.4. Identify the set of nondominated hyperintervals considering the current groups of hyperintervals from $q_\infty(k)$ up to $p(k)$. Subdivide those nondominated hyperintervals which satisfy inequality (18) and produce new trial points (or read the existing ones from the vertex database) according to Section 2.1. Set $k := k+1$, update all necessary indices.

Step 1.5. If the record hyperinterval is not the smallest one, i.e., if $p(k) < q_0(k)$, then go to Step 2 and execute the record improvement phase. Otherwise, go to Step 1 and repeat the exploration phase updating the value $f_{\min}^{\text{prec}}$.

Step 2 (*Record Improvement Phase*). Set $k := k+1$. Set the counter of iterations during the record improvement phase $k_l := 1$ and perform the following steps:

Step 2.1. Subdivide the record hyperinterval $D_{\min}(k)$ and produce a new trial point (or read the existing one from the vertex database) according to Section 2.1. Update hyperintervals indices and the record hyperinterval index if necessary.

Step 2.2. Increase the counter $k_l := k_l + 1$: check whether $k_l \leq N$. If this is the case, then go to Step 1 (perform a new exploration of large hyperintervals). Otherwise, go to Step 2.1 (continue the local exploration of the subregion near to the record point).

Let us now study convergence properties of the new method during minimization of the function $f(x)$ from (2)–(4) when the maximal allowed number of generated trial points $P_{\max}$ is equal to infinity. In this case, the algorithm does not stop (the number of iterations k goes to infinity) and an infinite sequence of trial points $\{x^{j(k)}\}$ is generated.

Definition 4. The convergence of an infinite sequence of trial points $\{x^{j(k)}\}$ generated by a global optimization method is called *everywhere dense* if for any point $x \in D$ and any $\delta > 0$ there exist an iteration number $k(\delta) \geq 1$ and a point $x' \in \{x^{j(k)}\}$, $k > k(\delta)$, such that $\|x - x'\| < \delta$.

Theorem 2. *The new algorithm manifests the everywhere dense convergence.*

Proof. Every subdivision of a hyperinterval by the partitioning scheme from Section 2.1 produces three new hyperintervals with the same volume equal to the third part of the volume of the subdivided hyperinterval and smaller main diagonals. Trial points generated by the new algorithm are at one of the vertices of each generated hyperinterval. Therefore, fixed a positive value of δ, it is sufficient to demonstrate that after a finite number of iterations $k(\delta)$, the largest hyperinterval of the current partition of the search domain D will have the length of its main diagonal smaller than δ. In this case, in δ-neighborhood of any point of D there will exist at least one trial point generated by the algorithm.

Let us fix an iteration k' of the method and consider the group $q_\infty(k')$ of the largest hyperintervals of the partition $\{D^{k'}\}$ on its two-dimensional graphic representation. This group is always taken into account when nondominated hyperintervals are looked for at the exploration phase of the algorithm. As it follows from Definition 3, a hyperinterval $D_t \in \{D^{k'}\}$ from this group with the corresponding smallest value F_t from (13) to (15), must be partitioned and substituted by three smaller hyperintervals at the current iteration of the algorithm because it is a nondominated hyperinterval and condition (18) is satisfied for it.

Since each group consists of a finite number of hyperintervals, after a sufficiently large number of iterations $k > k'$ all hyperintervals of the group $q_\infty(k)$ of the largest hyperintervals will be subdivided. The group index $q_\infty(k)$ will increase and the same procedure will be repeated with a new group of the largest hyperintervals, thus making the largest hyperintervals smaller and smaller.

It can be also noted that the record hyperinterval $D_{\min}$ is itself represented by a dot in the two-dimensional diagram of the current partition. It can be subdivided either separately during the record improvement phase, or as a nondominated hyperinterval during the exploration phase at which the satisfaction of condition (20) is not taken in consideration.

Thus, there exists a finite number $k(\delta)$ such that after executing $k(\delta)$ iterations of the algorithm the largest hyperinterval of the current partition $\{D^{k(\delta)}\}$ will have the main diagonal smaller than δ. □

To conclude the theoretical study of the new algorithm we would like to highlight that the usage of all possible estimates of the Lipschitz constant in its work leads to the convergence of the everywhere dense type. If the Lipschitz constant L (or its valid estimate) of the objective function $f(x)$ or the Lipschitz constant K (or its valid estimate) of the gradient $f'(x)$ can be used by a global optimization method, other types of convergence can be established for such an algorithm (see, e.g., methods from [2,9,10,15,36]).

4. Numerical results

In this Section, we present numerical results performed to compare the new algorithm with two methods belonging to the same class of the one-point-base partitioning methods: the DIRECT algorithm from [23] and its locally-biased modification DIRECT*l* from [21]. Both of them use the center-sampling partitioning strategy and work with a set of Lipschitz constants for the objective function $f(x)$ from (2). The implementation of these two methods (downloadable from http://www4.ncsu.edu/~ctk/SOFTWARE/DIRECTv204.tar.gz) has been used in all the experiments following the way of the multicriteria comparison proposed in [25].

In order to make easier the numerical comparison with the DIRECT-based algorithms, the value ξ from (18) was set as in the DIRECT method, i.e.,

$$\xi = \epsilon|f_{\min}(k)|, \quad \epsilon \geq 0. \tag{21}$$

The recommended value of $\epsilon = 10^{-4}$ (see [23,25]) was used in (21).

In accordance with [25], the global minimizer $x^* \in D$ was considered to be found when a method generated a trial point x' inside a hyperinterval with a vertex x^* and the volume smaller than the volume of the initial hyperinterval $D = [a, b]$ multiplied by an accuracy coefficient Δ, $0 < \Delta \leq 1$, i.e.,

$$|x'(i) - x^*(i)| \leq \sqrt[N]{\Delta}(b(i) - a(i)), \quad 1 \leq i \leq N, \tag{22}$$

where N is from (4). The algorithm was stopped either when the maximal number of trials $P_{\max}$ equal to 1 000 000 was reached, or when condition (22) was satisfied (see [25] for a discussion about different stopping criteria in global optimization methods).

In our numerical experiments we used the same test classes, each of 100 continuously differentiable functions, produced by the GKLS-generator (see [43]) as in [25]. Particularly, eight GKLS D-type classes of dimensions $N = 2, 3, 4$, and 5 have been considered. For each particular problem dimension N a 'simple' and a 'hard' classes have been taken for the comparison (see [25] for a detailed description of the classes).

For the convenience of the reader, we report here the four criteria introduced in [10,25] that were used to compare the methods. The following designations are required:

P_s—the number of trials performed by the method under consideration to solve the problem number s, $1 \leq s \leq 100$, of a fixed test class.

m_s—the number of hyperintervals generated to solve the problem s.

Criterion C1. Number of trials P_{s^*} required for a method to satisfy condition (22) for *all* 100 functions of a particular test class, i.e.,

$$P_{s^*} = \max_{1 \leq s \leq 100} P_s, \quad s^* = \arg \max_{1 \leq s \leq 100} P_s. \tag{23}$$

Criterion C2. The corresponding number of hyperintervals, m_{s^*}, generated by the method, where s^* is from (23).

Criterion C3. Average number of trials P_{avg} performed by the method during minimization of *all* 100 functions from a particular test class, i.e.,

$$P_{avg} = \frac{1}{100} \sum_{s=1}^{100} P_s. \tag{24}$$

Criterion C4. Number p (number q) of functions from a class for which DIRECT or DIRECT*l* executed less (more) function evaluations than the new algorithm. If P_s is the number of trials performed by the new algorithm and P'_s is the corresponding number of trials performed by a competing method, p and q are evaluated as follows

$$p = \sum_{s=1}^{100} \sigma'_s, \quad \sigma'_s = \begin{cases} 1, & P'_s < P_s, \\ 0, & \text{otherwise.} \end{cases} \tag{25}$$

$$q = \sum_{s=1}^{100} \sigma_s, \quad \sigma_s = \begin{cases} 1, & P_s < P'_s, \\ 0, & \text{otherwise.} \end{cases} \tag{26}$$

Results based on Criteria C1 and C2 are mainly influenced by minimization of the most difficult functions of a class. Criteria C3 and C4 deal with average data of a class. The number of generated hyperintervals (Criterion C2) provides an important characteristic of any partition algorithm for solving the problem (2)–(4). In some way, it corresponds to the qualitative

Table 1
Number of trial points for GKLS test functions (Criterion C1).

N	Δ	Class	50%			100%		
			DIRECT	DIRECT*l*	New	DIRECT	DIRECT*l*	New
2	10^{-4}	Simple	111	152	59	1159	2318	335
2	10^{-4}	Hard	1062	1328	182	3201	3414	1075
3	10^{-6}	Simple	386	591	362	12507	13309	2043
3	10^{-6}	Hard	1749	1967	416	>1000000 (4)	29233	2352
4	10^{-6}	Simple	4805	7194	2574	>1000000 (4)	118744	16976
4	10^{-6}	Hard	16114	33147	3773	>1000000 (7)	287857	20866
5	10^{-7}	Simple	1660	9246	1757	>1000000 (1)	178217	16300
5	10^{-7}	Hard	55092	126304	13662	>1000000 (16)	>1000000 (4)	88459

Table 2
Improvement obtained by the new algorithm in terms of Criterion C1.

N	Δ	Class	DIRECT/new	DIRECT*l*/new
2	10^{-4}	Simple	3.46	6.92
2	10^{-4}	Hard	2.98	3.18
3	10^{-6}	Simple	6.12	6.51
3	10^{-6}	Hard	>425.17	12.43
4	10^{-6}	Simple	>58.91	6.99
4	10^{-6}	Hard	>47.92	13.80
5	10^{-7}	Simple	>61.35	10.93
5	10^{-7}	Hard	>11.30	>11.30

Table 3
Number of hyperintervals for GKLS test functions (Criterion C2).

N	Δ	Class	50%			100%		
			DIRECT	DIRECT*l*	New	DIRECT	DIRECT*l*	New
2	10^{-4}	Simple	111	152	185	1159	2318	1137
2	10^{-4}	Hard	1062	1328	607	3201	3414	3993
3	10^{-6}	Simple	386	591	1867	12507	13309	12149
3	10^{-6}	Hard	1749	1967	2061	>1000000 (4)	29233	14357
4	10^{-6}	Simple	4805	7194	21635	>1000000 (4)	118744	186295
4	10^{-6}	Hard	16114	33147	33173	>1000000 (7)	287857	223263
5	10^{-7}	Simple	1660	9246	19823	>1000000 (1)	178217	255059
5	10^{-7}	Hard	55092	126304	169413	>1000000 (16)	>1000000 (4)	1592969

examination of the search domain D during the work of the method. The greater is this number, the more information about the behavior of the objective function is available and, therefore, the smaller is the risk to miss its global minimizer. Of course, algorithms should not generate many redundant hyperintervals since this slows down the search and is therefore a disadvantage of the method (see [25] for more details).

Results of numerical comparison of the methods with respect to Criteria C1 and C2 with eight GKLS test classes are shown in Tables 1–3. The accuracy coefficient Δ from (22) is given in the second column of the tables. Table 1 reports the maximal number of trials required for satisfying condition (22) for half of the functions of a particular class (columns "50%") and for all 100 function of the class (columns "100%"). The notation '>1 000 000 (j)' in Tables 1 and 3 means that after 1 000 000 function evaluations the method under consideration was not able to solve j problems. Table 2 represents the ratio between the maximal number of trials performed by DIRECT and DIRECT*l* with respect to the corresponding number of trials performed by the new algorithm. The numbers of generated hyperintervals (Criterion C2) are indicated in Table 3.

According to Tables 1 and 3, the new multidimensional algorithm requires much fewer trials than the other two methods to ensure a thorough examination of the search domain. Moreover, the advantage of the new method becomes even more pronounced as the problem dimension grows or the problem complexity increases.

In fact, on half of the test functions from each class (which were the most simple for each method with respect to the other functions of the class) the new algorithm already manifested a very good performance with respect to DIRECT and DIRECT*l* in terms of the number of generated trial points (see columns "50%" in Table 1). When all the functions were taken in consideration (and, consequently, difficult functions of the class were considered too), the number of trials produced by the new algorithm was much fewer in comparison with two other methods (see columns "100%" in Table 1), ensuring at the same time a substantial examination of the admissible domain (see Table 3).

Table 4
Average number of trial points for GKLS test functions (Criterion C3).

N	Δ	Class	DIRECT	DIRECT*l*	New	Improvement	
						DIRECT/new	DIRECT*l*/new
2	10^{-4}	Simple	198.89	292.79	97.22	2.06	3.01
2	10^{-4}	Hard	1063.78	1267.07	192.00	5.54	6.60
3	10^{-6}	Simple	1117.70	1785.73	491.28	2.28	3.63
3	10^{-6}	Hard	>42322.65	4858.93	618.32	>68.45	7.86
4	10^{-6}	Simple	>47282.89	18983.55	3675.84	>12.87	5.16
4	10^{-6}	Hard	>95708.25	68754.02	5524.77	>17.32	12.44
5	10^{-7}	Simple	>16057.46	16758.44	3759.05	>4.27	4.46
5	10^{-7}	Hard	>217215.58	>269064.35	22189.47	>9.79	>12.13

Table 5
Comparison between the new algorithm and DIRECT and DIRECT*l* in terms of Criterion C4.

N	Δ	Class	DIRECT:New	DIRECT*l*:new
2	10^{-4}	Simple	28:72	21:79
2	10^{-4}	Hard	15:85	16:84
3	10^{-6}	Simple	36:64	30:70
3	10^{-6}	Hard	19:81	17:83
4	10^{-6}	Simple	39:61	25:75
4	10^{-6}	Hard	14:86	16:84
5	10^{-7}	Simple	55:45	17:83
5	10^{-7}	Hard	26:74	20:80

Note also that maximal number of trials equal to 88 459 (see Table 1) required by the new method to solve all problems of the hard five-dimensional class is obtained on the function 5 of this class. If we use the new method with the one-point-based strategy starting from the point b rather than from the point a (see Section 2.1), the number of trials required by the new algorithm to solve this particular problem becomes equal to 15 238. Thus, some a priori knowledge on the objective function behavior can allow us to better select the vertex of the initial hyperinterval D in which the first trial will be executed and, therefore, to accelerate the search even more.

Table 4 reports the average number of trials performed during minimization of all 100 functions from the same GKLS classes (Criterion C3). The "Improvement" columns in these tables represent the ratios between the average numbers of trials performed by DIRECT and DIRECT*l* with respect to the corresponding numbers of trials performed by the new algorithm. The symbol '>' reflects the situation when not all functions of a class were successfully minimized by the method under consideration in the sense of condition (22). This means that the method stopped when $P_{\max}$ trials had been executed during minimization of several functions of this particular test class. In these cases, the value of $P_{\max}$ equal to 1 000 000 was used in calculations of the average value in (24), providing in such a way a lower estimate of the average. As can be seen from Table 4, the new method outperforms DIRECT and DIRECT*l* also on Criterion C3.

Finally, results of comparison between the new algorithm and its two competitors in terms of Criterion C4 are reported in Table 5. This table shows how often the new algorithm was able to minimize each of 100 functions of a class with a smaller number of trials with respect to DIRECT or DIRECT*l*. The notation 'p:q' means that among 100 functions of a particular test class there are p functions for which DIRECT (or DIRECT*l*) spent fewer function trials than the new algorithm and q functions for which the new algorithm generated fewer trial points with respect to DIRECT (or DIRECT*l*) (p and q are from (25) and (26), respectively). As a rule, the more hard objective functions are presented in a test class, the more pronounced becomes the advantage of the new algorithm on Criterion C4, as well.

As demonstrated by the results of the extensive numerical experiments performed, the usage of the gradient information together with the efficient partitioning strategy allows one to obtain a serious acceleration in comparison with the DIRECT-based methods on the studied classes of test problems.

References

[1] P.M. Pardalos, H.E. Romeijn, H. Tuy, Recent developments and trends in global optimization, J. Comput. Appl. Math. 124 (2000) 209–228.
[2] R.G. Strongin, Y.D. Sergeyev, Global Optimization with Non-Convex Constraints: Sequential and Parallel Algorithms, Kluwer Academic Publishers, Dordrecht, 2000.
[3] D. Trigiante (Ed.), Recent Trends in Numerical Analysis, Nova Science Publishers, Inc., New York, 2000.
[4] A.A. Zhigljavsky, A. Žilinskas, Stochastic Global Optimization, Springer, New York, 2008.
[5] C. Audet, P. Hansen, G. Savard (Eds.), Essays and Surveys in Global Optimization, in: GERAD 25th Anniversary, Springer-Verlag, New York, 2005.
[6] L. Dumas, B. Druez, N. Lecerf, A fully adaptive hybrid optimization of aircraft engine blades, J. Comput. Appl. Math. 232 (2009) 54–60.
[7] C. Luo, S. Zhang, C. Wang, Z. Jiang, A metamodel-assisted evolutionary algorithm for expensive optimization, J. Comput. Appl. Math. 236 (2011) 759–764.

[8] J. Mockus, A Set of Examples of Global and Discrete Optimization: Applications of Bayesian Heuristic Approach, Kluwer Academic Publishers, Dordrecht, 2000.
[9] J. Pintér, Global Optimization in Action (Continuous and Lipschitz Optimization: Algorithms, Implementations and Applications), Kluwer Academic Publishers, Dordrecht, 1996.
[10] Y.D. Sergeyev, D.E. Kvasov, Diagonal Global Optimization Methods, FizMatLit, Moscow, 2008, (In Russian).
[11] M. Gaviano, D. Lera, A global minimization algorithm for Lipschitz functions, Optim. Lett. 2 (2008) 1–13.
[12] Y. Nesterov, Introductory Lectures on Convex Optimization: A Basic Course, Kluwer Academic Publishers, Dordrecht, 2004.
[13] Y.G. Evtushenko, Numerical Optimization Techniques, in: Translations Series in Mathematics and Engineering, Springer-Verlag, Berlin, 1985.
[14] R. Horst, P.M. Pardalos (Eds.), Handbook of Global Optimization, volume 1, Kluwer Academic Publishers, Dordrecht, 1995.
[15] R. Horst, H. Tuy, Global Optimization—Deterministic Approaches, Springer-Verlag, Berlin, 1996.
[16] Y.G. Evtushenko, V.U. Malkova, A.A. Stanevichyus, Parallel global optimization of functions of several variables, Comput. Math. Math. Phys. 49 (2009) 246–260.
[17] D.E. Kvasov, Y.D. Sergeyev, Multidimensional global optimization algorithm based on adaptive diagonal curves, Comput. Math. Math. Phys. 43 (2003) 42–59.
[18] D.E. Kvasov, C. Pizzuti, Y.D. Sergeyev, Local tuning and partition strategies for diagonal GO methods, Numer. Math. 94 (2003) 93–106.
[19] Y.D. Sergeyev, An information global optimization algorithm with local tuning, SIAM J. Optim. 5 (1995) 858–870.
[20] D.E. Finkel, C.T. Kelley, Additive scaling and the DIRECT algorithm, J. Global Optim. 36 (2006) 597–608.
[21] J.M. Gablonsky, C.T. Kelley, A locally-biased form of the DIRECT algorithm, J. Global Optim. 21 (2001) 27–37.
[22] J. He, L.T. Watson, N. Ramakrishnan, C.A. Shaffer, A. Verstak, J. Jiang, K. Bae, W.H. Tranter, Dynamic data structures for a direct search algorithm, Comput. Optim. Appl. 23 (2002) 5–25.
[23] D.R. Jones, C.D. Perttunen, B.E. Stuckman, Lipschitzian optimization without the Lipschitz constant, J. Optim. Theory Appl. 79 (1993) 157–181.
[24] G. Liuzzi, S. Lucidi, V. Piccialli, A partition-based global optimization algorithm, J. Global Optim. 48 (2010) 113–128.
[25] Y.D. Sergeyev, D.E. Kvasov, Global search based on efficient diagonal partitions and a set of Lipschitz constants, SIAM J. Optim. 16 (2006) 910–937.
[26] D. Di Serafino, G. Liuzzi, V. Piccialli, F. Riccio, G. Toraldo, A modified DIviding RECTangles algorithm for a problem in astrophysics, J. Optim. Theory Appl. 151 (2011) 175–190.
[27] P.A. Graf, K. Kim, W.B. Jones, L. Wang, Surface passivation optimization using DIRECT, J. Comput. Phys. 224 (2007) 824–835.
[28] C.G. Moles, P. Mendes, J.R. Banga, Parameter estimation in biochemical pathways: a comparison of global optimization methods, Genome Res. 13 (2003) 2467–2474.
[29] T.D. Panning, L.T. Watson, N.A. Allen, K.C. Chen, C.A. Shaffer, J.J. Tyson, Deterministic parallel global parameter estimation for a model of the budding yeast cell cycle, J. Global Optim. 40 (2008) 719–738.
[30] D.E. Kvasov, Y.D. Sergeyev, A univariate global search working with a set of Lipschitz constants for the first derivative, Optim. Lett. 3 (2009) 303–318.
[31] Y.G. Evtushenko, M.A. Posypkin, An application of the nonuniform covering method to global optimization of mixed integer nonlinear problems, Comput. Math. Math. Phys. 51 (2011) 1286–1298.
[32] Y.D. Sergeyev, Efficient partition of N-dimensional intervals in the framework of one-point-based algorithms, J. Optim. Theory Appl. 124 (2005) 503–510.
[33] V.P. Gergel, A global optimization algorithm for multivariate function with Lipschitzian first derivatives, J. Global Optim. 10 (1997) 257–281.
[34] D. Lera, Y.D. Sergeyev, Lipschitz and Hölder global optimization using space-filling curves, Appl. Numer. Math. 60 (2010) 115–129.
[35] Y. Wu, L. Ozdamar, A. Kumar, TRIOPT: a triangulation-based partitioning algorithm for global optimization, J. Comput. Appl. Math. 177 (2005) 35–53.
[36] Y.D. Sergeyev, D.E. Kvasov, Lipschitz global optimization, in: J.J. Cochran (Ed.), in: Wiley Encyclopedia of Operations Research and Management Science, vol. 4, Wiley, New York, 2011, pp. 2812–2828.
[37] D.R. Jones, M. Schonlau, W.J. Welch, Efficient global optimization of expensive black-box functions, J. Global Optim. 13 (1998) 455–492.
[38] D. Lera, Y.D. Sergeyev, An information global minimization algorithm using the local improvement technique, J. Global Optim. 48 (2010) 99–112.
[39] W. Baritompa, Customizing methods for global optimization—a geometric viewpoint, J. Global Optim. 3 (1993) 193–212.
[40] Y.D. Sergeyev, Global one-dimensional optimization using smooth auxiliary functions, Math. Program. 81 (1998) 127–146.
[41] Y.D. Sergeyev, P. Daponte, D. Grimaldi, A. Molinaro, Two methods for solving optimization problems arising in electronic measurements and electrical engineering, SIAM J. Optim. 10 (1999) 1–21.
[42] L. Breiman, A. Cutler, A deterministic algorithm for global optimization, Math. Program. 58 (1993) 179–199.
[43] M. Gaviano, D. Lera, D.E. Kvasov, Y.D. Sergeyev, Algorithm 829: software for generation of classes of test functions with known local and global minima for global optimization, ACM Trans. Math. Software 29 (2003) 469–480.
[44] Y.D. Sergeyev, An efficient strategy for adaptive partition of N-dimensional intervals in the framework of diagonal algorithms, J. Optim. Theory Appl. 107 (2000) 145–168.
[45] J. Nocedal, S.J. Wright, Numerical Optimization, Springer-Verlag, Dordrecht, 1999.
[46] M. Gaviano, D. Lera, A complexity analysis of local search algorithms in global optimization, Optim. Methods Softw. 17 (2002) 113–127.