


# Parallel distributed block coordinate descent methods based on pairwise comparison oracle

Kota Matsui<sup>1</sup>  · Wataru Kumagai<sup>2</sup> · Takafumi Kanamori<sup>3</sup>

Received: 28 October 2015 / Accepted: 10 September 2016  
© Springer Science+Business Media New York 2016

**Abstract** This paper provides a block coordinate descent algorithm to solve unconstrained optimization problems. Our algorithm uses only pairwise comparison of function values, which tells us only the order of function values over two points, and does not require computation of a function value itself or a gradient. Our algorithm iterates two steps: the direction estimate step and the search step. In the direction estimate step, a Newton-type search direction is estimated through a block coordinate descent-based computation method with the pairwise comparison. In the search step, a numerical solution is updated along the estimated direction. The computation in the direction estimate step can be easily parallelized, and thus, the algorithm works efficiently to find the minimizer of the objective function. Also, we theoretically derive an upper bound of the convergence rate for our algorithm and show that our algorithm achieves the optimal query complexity for specific cases. In numerical experiments, we show that our method efficiently finds the optimal solution compared to some existing methods based on the pairwise comparison.

**Keywords** Derivative-free optimization · Pairwise comparison oracle · Block coordinate descent · Parallel computation

## 1 Introduction

Recently, demand for large-scale complex optimization is increasing in computational science, engineering and many other fields. In that kind of problems, there are many difficulties caused by noise in function evaluation, many tuning parameters and high computation cost. In

---

✉ Kota Matsui  
matsui.k@med.nagoya-u.ac.jp

<sup>1</sup> Nagoya University, Tsurumai-cho, Showa-ku, Nagoya 466-8550, Aichi, Japan

<sup>2</sup> Kanagawa University, 3-27-1, Rokkakubashi, Kanagawa-ku, Yokohama 221-8686, Kanagawa, Japan

<sup>3</sup> Nagoya University, Furo-cho, Chikusa-ku, Nagoya 464-8601, Japan

such cases, derivatives of the objective function are unavailable or computationally infeasible. These problems can be treated by the derivative-free optimization (DFO) methods.

DFO is the tool for optimization without derivative information of the objective function and constraints, and it has been widely studied for decades [5, 20]. There are roughly two approaches in DFO. The first one is first order methods with estimated gradient [8, 9] and the second one is zero-th order methods (or direct search methods) [1, 16]. A first order method is, for example, the usual gradient descent algorithm in which the gradient of the objective function is unbiasedly estimated. On the other hand, zero-th order methods do not use the true gradient. It uses only function values to estimate a descent direction of the objective function.

There is, however, a more restricted setting in which not only derivatives but also values of the objective function are unavailable or computationally infeasible. In such a situation, the so-called pairwise comparison oracle, that tells us an order of function values on two evaluation points, is used instead of derivatives and function evaluation [11, 16]. For example, the pairwise comparison is used in learning to rank to collect training samples to estimate the preference function of the ranking problems [15]. In decision making, finding the most preferred feasible solution from among the set of many alternatives is an important application of ranking methods using the pairwise comparison. Also, other type of information such as stochastic gradient-sign oracle has been studied [18].

*Related works* Nelder and Mead's downhill simplex method (shortly, Nelder–Mead method) [16] was proposed in an early study of algorithms based on the pairwise comparison of function values. In each iteration of the algorithm, a simplex that approximates the objective function is constructed according to a ranking of function values on sampled points. Then, the simplex receives four operations, namely, reflection, expansion, contraction and reduction in order to get close to the optimal solution. The Nelder–Mead method is a state-of-the-art in general optimization (e.g. “optim”, a general optimization function in R language, employs the Nelder–Mead method as a standard solver). Unfortunately, the convergence of the Nelder–Mead algorithm is theoretically guaranteed only in low dimensional problems [13]. In high dimensional problems, the Nelder–Mead algorithm works poorly as shown in [10].

Mesh adaptive direct search (MADS) [3, 6], another deterministic direct search algorithm, can treat non-smooth problems and hidden constraints. Each iteration of MADS consists of three procedures namely, the search, the poll, and the updates. The search explores a finite number of trial points on the *mesh*, constructed from a finite set of scaled directions (namely simplex gradients direction). The poll computes a descent direction from the generated trial points and the current solution is updated along the direction. MADS employs the passive-aggressive strategy, that is, if the candidate point  $y$  is not “better than” current solution  $x$  (represented by  $y \not\prec x$ ), the iteration is a failure and no update is done. Custódio et al. [7] analyze the properties of the simplex gradients that is used in some direct search methods that include MADS. The parallelization of MADS algorithm is considered in several ways e.g. parallel space decomposition [2].

Stochastic coordinate descent (SCD) algorithm is one of the most powerful algorithm for the (especially large scale) optimization problems. Richtárik and Takáč [19] provided a parallelization of SCD (PSCD) for the class of “partially separable” objective function that is, the objective function  $f$  can be represented as the sum of partial objective  $f_J$ :  $f(\mathbf{x}) = \sum_{J \in \{1, \dots, n\}} f_J$ . Here, the decision variable of  $f_J$  and  $f_{J'}$  do not intersect if  $J \cap J' = \emptyset$ .

Dueling bandits (DB) [21–23] is an online optimization problem where one can use only comparisons (called duels) between two points in a set with constraints. This problem is

**Table 1** Summary of the relationships between proposal and related works

| Algorithm   | Method        | Underlying setup | Required information | Parallelization |
|-------------|---------------|------------------|----------------------|-----------------|
| Nelder–Mead | Deterministic | Deterministic    | Pairwise comparison  |                 |
| MADS        | Deterministic | Deterministic    | Function value       | ✓               |
| PSCD        | Stochastic    | Deterministic    | Gradient             | ✓               |
| DB          | Stochastic    | Stochastic       | Pairwise comparison  |                 |
| PCSCD       | Stochastic    | Stochastic       | Pairwise comparison  |                 |
| Proposal    | Stochastic    | Stochastic       | Pairwise comparison  | ✓               |

Method: either the algorithm is deterministic or stochastic. Underlying setup: if the problem contains stochastic oracle or not. Required information: the information that is actually used in the algorithm. Parallelization: if the parallelization of the algorithm is considered or not. Our proposal has two characterizations, that is, the first is a parallelization of the PCSCD and the second is the PSCD under the restricted information

motivated by optimization of an information retrieval system (such as web search) to provide answers of queries that maximize user utility. Yue et al. [21] proposed a gradient descent type algorithm for this problem that uses stochastic pairwise comparisons. In the algorithm, update is taken along a random direction if the candidate point wins the current point by a duel.

The stochastic coordinate descent algorithm using only the noisy pairwise comparison (PCSCD) was proposed in [11]. Lower and upper bounds of the convergence rate were also presented in terms of the number of pairwise comparison of function values, i.e., query complexity. The algorithm iteratively solves one-dimensional optimization problems like the coordinate descent method. However, practical performance of the optimization algorithm was not studied in that work.

The relationships between proposal and related works are summarized in Table 1.

In this paper, we focus on optimization algorithms using the pairwise comparison oracle. In our algorithm, the convergence to the optimal solution is guaranteed, when the number of pairwise comparison tends to infinity. Our algorithm is regarded as a block coordinate descent method consisting of two steps: the direction estimate step and search step. In the direction estimate step, the search direction is determined. In the search step, the current solution is updated along the search direction with an appropriate step length. In our algorithm, the direction estimate step is easily parallelized. Therefore, it is expected that our algorithm effectively works even in large-scale optimization problems.

Let us summarize the contributions presented in this paper.

1. We propose a block coordinate descent algorithm based on the pairwise comparison oracle, and point out that the algorithm is easily parallelized.
2. We derive an upper bound of the convergence rate in terms of the number of pairwise comparison of function values, i.e., query complexity. Moreover, we show that the query complexity of our algorithm is optimal for specific cases.
3. We show that our proposal is more efficient than the state-of-the-art methods using the same information (i.e. pairwise comparison oracle) through numerical experiments.

The rest of the paper is organized as follows. In Sect. 2, we explain the problem setup and give some definitions. Section 3 is devoted to the main results. The convergence properties and query complexity of our algorithm are shown in the section. In Sect. 4, numerical examples are reported. Finally in Sect. 5, we conclude the paper with the discussion on future works. All proofs of theoretical results are found in the “Appendix”.

## 2 Preliminaries

In this section, we introduce the problem setup and prepare some definitions and notations used throughout the paper. A function  $f: \mathbb{R}^n \rightarrow \mathbb{R}$  is said to be  $\sigma$ -strongly convex on  $\mathbb{R}^n$  for a positive constant  $\sigma$ , if for all  $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ , the inequality

$$f(\mathbf{y}) \geq f(\mathbf{x}) + \nabla f(\mathbf{x})^\top (\mathbf{y} - \mathbf{x}) + \frac{\sigma}{2} \|\mathbf{x} - \mathbf{y}\|^2 \quad (1)$$

holds, where  $\nabla f(\mathbf{x})$  and  $\|\cdot\|$  denote the gradient of  $f$  at  $\mathbf{x}$  and the Euclidean norm, respectively. The function  $f$  is  $L$ -strongly smooth for a positive constant  $L$ , if  $\|\nabla f(\mathbf{x}) - \nabla f(\mathbf{y})\| \leq L\|\mathbf{x} - \mathbf{y}\|$  holds for all  $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ . The gradient  $\nabla f(\mathbf{x})$  of the  $L$ -strongly smooth function  $f$  is referred to as  $L$ -Lipschitz gradient. The class of  $\sigma$ -strongly convex and  $L$ -strongly smooth functions on  $\mathbb{R}^n$  is denoted as  $\mathcal{F}_{\sigma,L}(\mathbb{R}^n)$ . In the convergence analysis, we focus mainly on the optimization of objective functions in  $\mathcal{F}_{\sigma,L}(\mathbb{R}^n)$ .

We consider the following pairwise comparison oracle defined in [11].

**Definition 1** (*Pairwise comparison oracle*) The stochastic pairwise comparison (PC) oracle is a binary valued random variable  $O_f: \mathbb{R}^n \times \mathbb{R}^n \rightarrow \{-1, +1\}$  defined as

$$\Pr[O_f(\mathbf{x}, \mathbf{y}) = \text{sign}\{f(\mathbf{y}) - f(\mathbf{x})\}] \geq \frac{1}{2} + \min\{\delta_0, \mu|f(\mathbf{y}) - f(\mathbf{x})|^{\kappa-1}\}, \quad (2)$$

where  $0 < \delta_0 \leq 1/2$ ,  $\mu > 0$  and  $\kappa \geq 1$ . For  $\kappa = 1$ , without loss of generality  $\mu \leq \delta_0 \leq 1/2$  is assumed.

When the equality

$$\Pr[O_f(\mathbf{x}, \mathbf{y}) = \text{sign}\{f(\mathbf{y}) - f(\mathbf{x})\}] = 1 \quad (3)$$

is satisfied for all  $\mathbf{x}$  and  $\mathbf{y}$ , we call  $O_f$  the deterministic PC oracle.

For  $\kappa = 1$ , the probability in (2) is not affected by the difference  $|f(\mathbf{y}) - f(\mathbf{x})|$ , and hence, the probability for the output of the PC oracle is not changed under any monotone transformation of  $f$ .

In [11], Jamieson et al. derived lower and upper bounds of convergence rate of an optimization algorithm using the stochastic PC oracle. The algorithm is referred to as the original PC algorithm in the present paper.

Under above preparations, our purpose is to find the minimizer  $\mathbf{x}^*$  of the objective function  $f(\mathbf{x})$  in  $\mathcal{F}_{\sigma,L}(\mathbb{R}^n)$  by using PC oracle. In the following section, we provide a DFO algorithm in order to solve the optimization problem and consider the convergence properties including query complexity.

## 3 Main results

### 3.1 Algorithm

In Algorithm 1, we propose a DFO algorithm based on the PC oracle. In our algorithm,  $m$  coordinates out of  $n$  elements are updated in each iteration to efficiently cope with high dimensional problems. Algorithm 1 is referred to as BlockCD[ $n, m$ ]. The original PC algorithm is recovered by setting  $m = 1$ . The PC oracle is used in the line search algorithm to solve one-dimensional optimization problems; the detailed line search algorithm is shown in Algorithm 2. The conditional statements in Update step are necessary to guarantee the monotone decrease of  $f(\mathbf{x}_t)$  (see also proof of Theorem 1).

**Algorithm 1** Block coordinate descent using PC oracle (BlockCD[ $n, m$ ])

**Input:** initial point  $\mathbf{x}_0 \in \mathbb{R}^n$ , and accuracy in line search  $\eta > 0$ .

**Initialize:** set  $t = 0$ .

**repeat**

Choose  $m$  coordinates  $i_1, \dots, i_m$  out of  $n$  coordinates according to the uniform distribution.

**(Direction estimate step)**

[Step D-1] Solve the one-dimensional optimization problems

$$\min_{\alpha \in \mathbb{R}} f(\mathbf{x}_t + \alpha \mathbf{e}_{i_k}), \quad k = 1, \dots, m, \quad (4)$$

within the accuracy  $\eta/2$  using the PC-based line search algorithm shown in Algorithm 2, where  $\mathbf{e}_i$  denotes the  $i$ th unit basis vector.

Then, obtain the numerical solutions  $\alpha_{t,i_k}, k = 1, \dots, m$ .

[Step D-2] Set  $\mathbf{d}_t = \sum_{k=1}^m \alpha_{t,i_k} \mathbf{e}_{i_k}$ .

If  $\mathbf{d}_t$  is the zero vector, set  $\mathbf{d}_t \leftarrow \frac{\eta}{2} \mathbf{e}_{i_1}$ .

**(Search step)**

[Step S-1] Apply Algorithm 2 to obtain a numerical solution  $\beta_t$  of

$$\min_{\beta} f(\mathbf{x}_t + \beta \mathbf{d}_t / \|\mathbf{d}_t\|)$$

within the accuracy  $\eta$ .

**(Update step)**

**if**  $f(\mathbf{x}_t) \geq f(\mathbf{x}_t + \beta_t \mathbf{d}_t / \|\mathbf{d}_t\|)$  **then**

$\mathbf{x}_{t+1} = \mathbf{x}_t + \beta_t \mathbf{d}_t / \|\mathbf{d}_t\|$ ;  $t \leftarrow t + 1$ .

**else**

$\mathbf{x}_{t+1} = \mathbf{x}_t$ ;  $t \leftarrow t + 1$ .

**end if**

**until** A stopping criterion is satisfied.

**Output:**  $\mathbf{x}_t$

For  $m = n$ , the search direction  $\mathbf{d}_t$  in Algorithm 1 approximates that of a modified Newton method [14, Chap. 10], as shown below. In Step D-1 of the algorithm, one-dimensional optimization problems (4) are solved.

Let  $\alpha_{t,i}^*$  be the optimal solution of (4) with  $i_k = i$ . Then,  $\alpha_{t,i}^*$  will be close to the numerical solution  $\alpha_{t,i}$ . The Taylor expansion of the objective function leads to

$$f(\mathbf{x}_t + \alpha \mathbf{e}_i) = f(\mathbf{x}_t) + \alpha \mathbf{e}_i^\top \nabla f(\mathbf{x}_t) + \frac{\alpha^2}{2} \mathbf{e}_i^\top \nabla^2 f(\mathbf{x}_t) \mathbf{e}_i + o(\alpha^2),$$

where  $\nabla^2 f(\mathbf{x}_t)$  is the Hessian matrix of  $f$  at  $\mathbf{x}_t$ . When the point  $\mathbf{x}_t$  is close to the optimal solution of  $f(\mathbf{x})$ , the optimal parameter  $\alpha_{t,i}^*$  will be close to zero, implying that the higher order term  $o(\alpha^2)$  in the above is negligible. Hence,  $\alpha_{t,i}^*$  is approximated by the optimal solution of the quadratic approximation, i.e.,  $-(\nabla f(\mathbf{x}_t))_i / (\nabla^2 f(\mathbf{x}_t))_{ii}$ . As a result, the search direction in BlockCD[ $n, n$ ] approximates  $-(\text{diag}(\nabla^2 f(\mathbf{x}_t)))^{-1} \nabla f(\mathbf{x}_t)$ , where  $\text{diag}(A)$  denotes the diagonal matrix, the diagonal elements of which are those of the square matrix  $A$ . In the modified Newton method, the Hessian matrix in the Newton method is replaced with a positive definite matrix to reduce the computation cost. Using only the diagonal part of the Hessian matrix is a popular choice in the modified Newton method.

Figure 1 demonstrates an example of the optimization process of both the original PC algorithm and our algorithm. The original PC algorithm updates the numerical solution along a randomly chosen coordinate in each iteration. Hence, many iterations are required to get close to the optimal solution. On the other hand, in our algorithm, a solution can move along an oblique direction. Therefore, our algorithm can get close to the optimal solution with less

**Algorithm 2** line search algorithm using PC oracle [11]

**Input:** current solution  $\mathbf{x}_t \in \mathbb{R}^n$ , search direction  $\mathbf{d} \in \mathbb{R}^n$  and accuracy in line search  $\eta > 0$ .

**Initialize:** set  $\alpha_0 = 0$ ,  $\alpha_0^+ = \alpha_0 + 1$ ,  $\alpha_0^- = \alpha_0 - 1$ ,  $k = 0$ .

**[Step1]**

**if**  $O_f(\mathbf{x}_t, \mathbf{x}_t + \alpha_0^+ \mathbf{d}) > 0$  and  $O_f(\mathbf{x}_t, \mathbf{x}_t + \alpha_0^- \mathbf{d}) < 0$  **then**

$\alpha_0^+ \leftarrow 0$

**else if**  $O_f(\mathbf{x}_t, \mathbf{x}_t + \alpha_0^+ \mathbf{d}) < 0$  and  $O_f(\mathbf{x}_t, \mathbf{x}_t + \alpha_0^- \mathbf{d}) > 0$  **then**

$\alpha_0^- \leftarrow 0$

**end if**

**[Step2]** (double-sign corresponds)

**while**  $O_f(\mathbf{x}_t, \mathbf{x}_t + \alpha_k^\pm \mathbf{d}) < 0$  **do**

$\alpha_{k+1}^\pm \leftarrow 2\alpha_k^\pm$ ,  $k \leftarrow k + 1$

**end while**

**[Step3]**

**while**  $|\alpha_k^+ - \alpha_k^-| > \eta/2$  **do**

**if**  $O_f(\mathbf{x}_t + \alpha_k \mathbf{d}, \mathbf{x}_t + \frac{1}{2}(\alpha_k + \alpha_k^+) \mathbf{d}) < 0$  **then**

$\alpha_{k+1} \leftarrow \frac{1}{2}(\alpha_k + \alpha_k^+)$ ,  $\alpha_{k+1}^+ \leftarrow \alpha_k^+$ ,  $\alpha_{k+1}^- \leftarrow \alpha_k$

**else if**  $O_f(\mathbf{x}_t + \alpha_k \mathbf{d}, \mathbf{x}_t + \frac{1}{2}(\alpha_k + \alpha_k^-) \mathbf{d}) < 0$  **then**

$\alpha_{k+1} \leftarrow \frac{1}{2}(\alpha_k + \alpha_k^-)$ ,  $\alpha_{k+1}^- \leftarrow \alpha_k^-$ ,  $\alpha_{k+1}^+ \leftarrow \alpha_k$

**else**

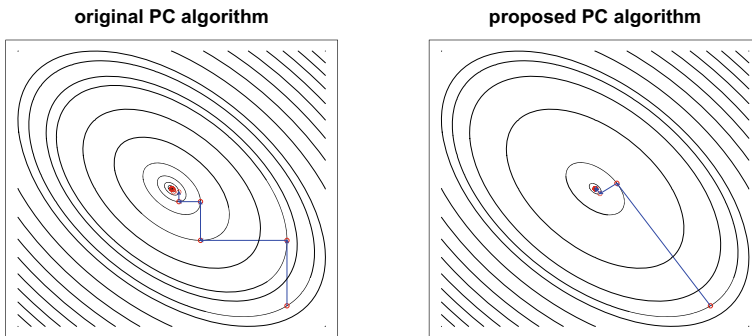
(double-sign corresponds)

$\alpha_{k+1} \leftarrow \alpha_k$ ,  $\alpha_{k+1}^\pm \leftarrow \frac{1}{2}(\alpha_k + \alpha_k^\pm)$

**end if**

**end while**

**Output:**  $\alpha_t$



**Fig. 1** A behavior of the algorithms on the contour of the quadratic objective function  $x_1^2 + x_2^2 + x_1 x_2$  with same initialization. *Left panel* Jamieson et al.'s original PC algorithm. *Right panel* proposed algorithm

iterations than the original PC algorithm. It should be noted here that although our algorithm requires higher computational costs in one iteration compared to original PC algorithm, parallelizability of direction estimate step reduces the practical computation time and our algorithm as a whole can efficiently find the optimal solution. The detail of parallelization of BlockCD $[n, m]$  is described in Sect. 4.

### 3.2 Convergence properties of our algorithm under deterministic oracle

We now provide an upper bound of the convergence rate of our algorithm using the deterministic PC oracle (3).

Let us denote the minimizer of  $f$  as  $\mathbf{x}^*$ .

**Theorem 1** Suppose  $f \in \mathcal{F}_{\sigma,L}(\mathbb{R}^n)$ , and set  $\varepsilon > 0$  arbitrarily. Define  $\gamma$  and  $\eta$  as

$$\gamma = \frac{\sigma/L}{52} \left( \frac{1 - \sqrt{1 - \sigma/L}}{1 + \sqrt{1 - \sigma/L}} \right)^2, \quad \eta = \sqrt{\frac{\sigma \varepsilon}{8nL^2(1 + n/m\gamma)}}. \quad (5)$$

Let us define  $T_0$  be

$$T_0 = \left\lceil \frac{n}{m\gamma} \log \frac{(f(\mathbf{x}_0) - f(\mathbf{x}^*)) \left(1 + \frac{n}{m\gamma}\right)}{\varepsilon} \right\rceil. \quad (6)$$

For  $T \geq T_0$ , we have  $\mathbb{E}[f(\mathbf{x}_T) - f(\mathbf{x}^*)] \leq \varepsilon$ , where the expectation is taken with respect to the random choice of coordinates  $i_1, \dots, i_k$  to be updated in BlockCD $[n, m]$  with the accuracy  $\eta$ .

The proof of Theorem 1 is given in “Appendix 1”. Note that any monotone transformation of the objective function does not affect the output of the deterministic PC oracle. Hence, the theorem is applicable for the composite of any function  $f$  and a non-decreasing function  $g$ ,  $g \circ f$ , if  $g \circ f \in \mathcal{F}_{\sigma,L}(\mathbb{R}^n)$ .

**Remark 1** As shown in (6), the number of iterations to achieve the accuracy  $\varepsilon$  is of the order  $T_0 = \tilde{O}(\lambda^3 \log(1/\varepsilon))$  for fixed  $n$  and  $m$ , where  $\lambda$  denotes the condition number of the Hessian matrix,  $\lambda = L/\sigma$ . On the other hand, an upper bound of the iteration number for non-linear optimization algorithms such as the steepest descent method is given by  $O(\lambda \log(1/\varepsilon))$  [14, Chap. 8.6.3]. The cubic order of  $\lambda$  in our method comes from the worst-case analysis of the numerical error in the search direction  $\mathbf{d}_t$  caused by multiple inexact line searches. In the standard setup of non-linear optimization problems, such a large numerical error in the search direction is not taken into account.

**Remark 2** Theorem 1 says that for arbitrary  $\varepsilon$ , if we take the tolerance  $\eta$ , of the line search as in (5), it is guaranteed that the average difference between optimal solution and current solution is smaller than  $\varepsilon$ . In contrast, if we take  $\eta$  arbitrary, it can be guaranteed that

$$\mathbb{E}[f(\mathbf{x}_T) - f(\mathbf{x}^*)] \leq \frac{8nL^2\eta^2 \left(1 + \frac{n}{m\gamma}\right)}{\sigma} \quad (7)$$

for  $T \geq T_0$ , where

$$T_0 = \left\lceil \frac{n}{m\gamma} \log \frac{(f(\mathbf{x}_0) - f(\mathbf{x}^*))\sigma}{8nL^2\eta^2} \right\rceil.$$

Here, when we get  $\eta$  small, the average difference compared with the optimal function value quadratically decreases with respect to  $\eta$  while the number  $T_0$  of total queries logarithmically increases at most. Thus, it means that we can get tight bound with only slight increment of total queries if we set the tolerance of a line search small.

**Remark 3** To the best of our knowledge, it is still an open problem to determine an appropriate stopping criterion of the algorithm under the pairwise comparison framework. The first order condition,  $\|\nabla f(\mathbf{x}_T)\| < \text{tolerance}$ , or the difference of function values,  $|f(\mathbf{x}_T) - f(\mathbf{x}_{T-1})| < \text{tolerance}$ , can not be employed under the pairwise comparison oracle because function values and gradients are not available. Yue et al. [21] or Jamieson et al. [11] also do not explicitly describe about the stopping criterion. In practice, we employ the norm of the difference of iterative solutions,  $\|\mathbf{x}_{t+1} - \mathbf{x}_t\| < \text{tolerance}$ .

Let us consider the convergence accuracy of the numerical solution  $\mathbf{x}_T$ . From the strong convexity (1), the inequality

$$\|\mathbf{x} - \mathbf{x}^*\|^2 \leq \frac{2}{\sigma} (f(\mathbf{x}) - f(\mathbf{x}^*))$$

holds. Thus, for  $T \geq T_0$ , we have

$$\mathbb{E}[\|\mathbf{x}_T - \mathbf{x}^*\|^2] \leq \mathbb{E}[\|\mathbf{x}_T - \mathbf{x}^*\|^2] \leq \frac{2}{\sigma} \varepsilon = 16n \left(\frac{L}{\sigma}\right)^2 \left(1 + \frac{n}{m\gamma}\right) \eta^2.$$

### 3.3 Query complexity

Let  $\hat{\mathbf{x}}_Q$  be the output of BlockCD[ $n, m$ ] after  $Q$  pairwise comparison queries. To solve the one-dimensional optimization problem within the accuracy  $\eta/2$ , the sufficient number of the call of PC-oracle is

$$K_0 = \left\lceil 2 \log_2 \frac{2^{10} L \Delta_0}{\sigma^2 \eta^2} \right\rceil = \left\lceil \frac{2}{\log 2} \log \frac{2^{13} (L/\sigma)^3 \Delta_0 \left(1 + \frac{n}{m\gamma}\right)}{\varepsilon} \right\rceil, \quad (8)$$

as shown in [11] where  $\Delta_0 = f(\mathbf{x}_0) - f(\mathbf{x}^*)$ . Hence, if the inequality  $Q \geq T_0 K_0 (m + 2)$  holds, Theorem 1 assures that the numerical solution  $\hat{\mathbf{x}}_Q$  based on  $Q$  queries satisfies

$$\mathbb{E}[f(\hat{\mathbf{x}}_Q) - f(\mathbf{x}^*)] \leq \varepsilon.$$

The following Corollary shows the query complexity of Algorithm 1.

**Corollary 1** For any  $Q$  such that

$$\frac{n}{c_0^2} \frac{m+2}{m\gamma} \left[ \log \left( 2^{13.5} \left(\frac{L}{\sigma}\right)^3 n \Delta_0 \left(1 + \frac{n}{m\gamma}\right) \right) \right]_+^2 \leq Q,$$

we have

$$\mathbb{E}[f(\mathbf{x}_Q) - f(\mathbf{x}^*)] \leq \exp \left\{ -c_0 \sqrt{\gamma \times \frac{m}{m+2} \times \frac{Q}{n}} \right\} \quad (9)$$

The proof of Corollary 1 is given in “Appendix 2”.

**Remark 4**  $\gamma = O((\sigma/L)^3)$  is regarded as the reciprocal of the condition number for the objective function.

**Remark 5** It is also guaranteed that there exists a constant  $C > 0$  such that for any  $Q$  (even if the assumption in Corollary 1 is not held),

$$\mathbb{E}[f(\mathbf{x}_Q) - f(\mathbf{x}^*)] \leq C \exp \left\{ -c_0 \sqrt{\gamma \times \frac{m}{m+2} \times \frac{Q}{n}} \right\} \quad (10)$$

holds.

The above bound is of the same order of the convergence rate for the original PC algorithm up to polylog factors. On the other hand, a lower bound presented in [11] is of order  $e^{-cQ/n}$  with a positive constant  $c$  up to polylog factors, when the PC oracle with  $\kappa = 1$  is used.



**Algorithm 3** Repeated querying subroutine ([11, 12])

---

**Input:**  $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n, \delta > 0$ .  
**Initialize:** set  $n_0 = 1$  and draw a stochastic PC oracle  $O_f(\mathbf{x}, \mathbf{y})$ .  
**for**  $k = 0, 1, \dots$  **do**  
    $p_k$  = frequency of +1 in all draw of  $O_f(\mathbf{x}, \mathbf{y})$  so far  
    $I_k = \left[ p_k - \sqrt{\frac{(k+1)\log(2/\delta)}{2^k}}, p_k + \sqrt{\frac{(k+1)\log(2/\delta)}{2^k}} \right]$   
   **if**  $\frac{1}{2} \notin I_k$  **then**  
      **break**  
   **else**  
      draw  $O_f(\mathbf{x}, \mathbf{y})$   $n_k$  more times, and set  $n_{k+1} = 2n_k$ .  
   **end if**  
**end for**  
**if**  $p_k + \sqrt{\frac{(k+1)\log(2/\delta)}{2^k}} \leq \frac{1}{2}$  **then**  
   **return** -1  
**else**  
   **return** +1  
**end if**

---

In Theorem 1, it is assumed that the objective function is strongly convex and gradient Lipschitz. In a realistic situation, we usually do not have the knowledge of the class parameters  $\sigma$  and  $L$  of the unknown objective function. Moreover, strong convexity and gradient Lipschitzness on the whole space  $\mathbb{R}^n$  is too strong.

In the following corollary, we relax the assumption in Theorem 1 and prove the convergence property of our algorithm without explicitly imposing strong convexity and strong smoothness.

**Corollary 2** *Let  $f: \mathbb{R}^n \rightarrow \mathbb{R}$  be a twice continuously differentiable convex function with non-degenerate Hessian on  $\mathbb{R}^n$  and have the minimizer  $\mathbf{x}^*$ . Then, there is a constant  $c$  and sufficiently small  $\eta$ ,<sup>1</sup> such that the output  $\hat{\mathbf{x}}_Q$  of BlockCD[n, m] satisfies (9).*

The proof of Corollary 2 is given in “Appendix 3”.

### 3.4 Generalization to stochastic pairwise comparison oracle

In this subsection, we generalize the results described in the previous section to the stochastic setting. Our strategy is to come down to the deterministic situation since for the deterministic PC oracle, the monotone decreasing of function value is guaranteed. As such, in Algorithm 1, the decrease of function value is guaranteed in high probability even if the deterministic PC oracle is replaced with the stochastic PC oracle. More specifically, for stochastic PC oracle, we derive the number of queries that is required to obtain the correct PC (i.e. deterministic PC) in high probability. Then, the total number of query for stochastic PC oracle is computed by the total number of query for deterministic PC oracle times the above number of queries.

In Algorithm 3, the query  $O_f(\mathbf{x}, \mathbf{y})$  is repeated under the stochastic PC oracle. The reliability of line search algorithm based on stochastic PC oracle (2) was investigated by [11, 12].

---

<sup>1</sup> “Sufficiently” means that  $\eta$  is smaller than or equal to the quantity of the right-hand-side of (5). Although the quantity can not be explicitly computed (since  $L$  and  $\sigma$  are unknown), we can achieve the order optimal by taking  $\eta$  smaller and smaller.

**Lemma 1** [11, 12] For any  $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$  with  $p = \Pr[O_f(\mathbf{x}, \mathbf{y}) = \text{sign}\{f(\mathbf{y}) - f(\mathbf{x})\}]$ , the repeated querying subroutine in Algorithm 3 correctly identifies the sign of  $\mathbb{E}[O_f(\mathbf{x}, \mathbf{y})]$  with probability  $1 - \delta$ , and requests no more than

$$\frac{\log 2/\delta}{4|1/2 - p|^2} \log_2 \left( \frac{\log 2/\delta}{4|1/2 - p|^2} \right) \quad (11)$$

queries.

It should be noted here that, in this paper,  $\text{sign}\{\mathbb{E}[O_f(\mathbf{x}, \mathbf{y})]\} = \text{sign}\{f(\mathbf{y}) - f(\mathbf{x})\}$  always holds because  $p > 1/2$  from (2). In [11], a modified line search algorithm using a ternary search instead of bisection search was proposed so that  $|1/2 - p|$  in Lemma 1 is lower bounded by  $\mu \left( \frac{\sigma \eta^2}{18} \right)^{\kappa-1}$  for arbitrary  $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$  where  $\eta$  is an accuracy of line search computed by (5) and  $\mu$  and  $\kappa$  are oracle parameters defined in (2).

Then one can find that, the total number of queries required to estimate the correct sign of  $\mathbb{E}[O_f(\mathbf{x}, \mathbf{y})]$  for arbitrary  $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$  with probability more than  $1 - \delta$ , is at most

$$\begin{aligned} Q(\delta) &= \frac{\log 2/\delta}{4 \left( \mu (\sigma \eta^2/18)^{\kappa-1} \right)^2} \log_2 \left( \frac{\log 2/\delta}{4 \left( \mu (\sigma \eta^2/18)^{\kappa-1} \right)^2} \right) \\ &= O \left( \log \frac{1}{\delta} \cdot \log \log \frac{1}{\delta} \right). \end{aligned}$$

Intuitively,  $Q(\delta)$  is the number of queries that is required in order to treat the stochastic PC oracle as the deterministic PC oracle. Let  $\delta'$  be a probability parameter such that we want to guarantee  $\mathbb{E}[f(\hat{\mathbf{x}}_Q) - f(\mathbf{x}^*)] \leq \varepsilon$  with probability more than  $1 - \delta'$  under stochastic PC oracle. Note that if we call  $J_0 Q(\delta')$  stochastic oracles, we can guarantee  $\mathbb{E}[f(\hat{\mathbf{x}}_Q) - f(\mathbf{x}^*)] \leq \varepsilon$  with probability  $(1 - \delta')^{J_0}$ , where

$$J_0 = T_0 K_0 (m + 2) \quad (12)$$

and  $T_0$  and  $K_0$  are defined in (19). This implies that if we call  $J_0 Q(\delta'/J_0)$  stochastic oracle, we can guarantee  $\mathbb{E}[f(\hat{\mathbf{x}}_Q) - f(\mathbf{x}^*)] \leq \varepsilon$  with probability  $(1 - \delta'/J_0)^{J_0}$  that is greater than  $1 - \delta'$ . That is, if we call

$$J_0 Q \left( \frac{\delta'}{J_0} \right) \approx O \left( J_0 \log \frac{J_0}{\delta'} \right) = \tilde{O} \left( J_0 \log \frac{1}{\delta'} \right) \quad (13)$$

stochastic oracle, we can guarantee  $\mathbb{E}[f(\hat{\mathbf{x}}_Q) - f(\mathbf{x}^*)] \leq \varepsilon$  with probability more than  $1 - \delta'$ . Hence, we have to take  $\delta$  in Lemma 1 such that

$$\delta \leq \frac{\delta'}{J_0} \quad (14)$$

to achieve the accuracy  $\varepsilon$ .

Then, if we take  $\delta$  which satisfies (14), it is sufficient to call

$$Q \geq J_0 Q(\delta) \quad (15)$$

stochastic oracle for guaranteeing  $\mathbb{E}[f(\hat{\mathbf{x}}_Q) - f(\mathbf{x}^*)] \leq \varepsilon$  with probability  $1 - \delta'$ .

Using the relation (5) between  $\eta$  and  $\varepsilon$  and solve the inequality (15) with respect to  $\varepsilon$ , we have the following upper bounds for stochastic setting:

$$\mathbb{E}[f(\hat{\mathbf{x}}_Q) - f(\mathbf{x}^*)] \leq \begin{cases} \exp\left\{-\frac{c_1}{\log n} \sqrt{\frac{Q}{n}}\right\}, & \kappa = 1, \\ c_2 \frac{n^2}{m} \left(\frac{n}{Q}\right)^{1/(2\kappa-2)}, & \kappa > 1, \end{cases} \quad (16)$$

where  $c_1$  and  $c_2$  are constant depending on oracle parameters,  $L/\sigma$  and  $f(\mathbf{x}_0) - f(\mathbf{x}^*)$  as well as  $1/\delta$  poly-logarithmically. If  $m$  and  $n$  are of the same order in the case of  $\kappa > 1$ , the bound (16) coincides with that shown in Theorem 2 of [11].

Note that our upper bound in (16) is optimal with respect to the number of queries  $Q$  when  $\kappa > 1$ . This result comes from the lower bound derived by Theorem 2 [11].

**Theorem 2** (Theorem 1 in [11]) *For  $n \geq 8$  and sufficiently large  $Q$ ,*

$$\inf_{\hat{\mathbf{x}}_Q} \sup_{f \in \mathcal{F}_{\sigma, L}(\mathbb{R}^n)} \mathbb{E}[f(\hat{\mathbf{x}}_Q) - f(\mathbf{x}_f^*)] \geq \begin{cases} C_1 \exp\{-C_2 \frac{Q}{n}\} & \text{if } \kappa = 1 \\ C_3 \left(\frac{n}{Q}\right)^{1/2(\kappa-1)} & \text{if } \kappa > 1 \end{cases} \quad (17)$$

where the constants  $C_1, C_2, C_3$  depend on the oracle and function class parameters.

Although Jamieson et al. also achieved the optimal rate with respect to the number of queries by PCSCD algorithm when  $\kappa > 1$  (Theorem 2 in [11]), our proposal, BlockCD algorithm, practically outperforms PCSCD algorithm by an effect of parallelization. We explain this in Sect. 4.2 in detail.

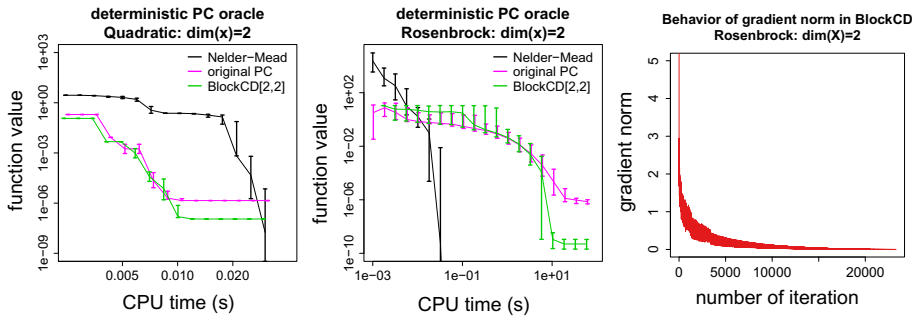
**Remark 6** In the case of  $\kappa > 1$ , the upper bound of the query complexity (16) with  $m = 1$  is of the order  $O(n^2(n/Q)^{1/(2\kappa-2)})$ , which is larger than the upper bound  $O(n(n/Q)^{1/(2\kappa-2)})$  presented in Theorem 2 of [11]. In our algorithm, not only the step length  $\beta$  but also the search direction  $\mathbf{d}_t$  is affected by numerical errors in the line search to the directions of  $\mathbf{e}_{i_k}$ ,  $k = 1, \dots, m$ . Though (16) provides the upper bound for any  $m = 1, \dots, n$ , that is not necessarily tight for a small  $m$ . A technical point for  $m = 1$  is explained in Remark 7 in “Appendix 1”. On the other hand, (16) with  $m = O(n)$  is  $O(n(n/Q)^{1/(2\kappa-2)})$ , which is of the same order of the original PC algorithm. Thus, the parallel implementation of the multiple line searches will be of great help to reducing the computation time, as will be shown in Sect. 4.

## 4 Numerical experiments

We present numerical experiments in which the proposed method in Algorithm 1 was compared with the Nelder–Mead algorithm [16], dueling bandit gradient descent method [21], and original PC algorithm [11], i.e., BlockCD[ $n, 1$ ] of Algorithm 1. Here, the PC oracle was used in all the optimization algorithms.

### 4.1 Existing methods and preliminary experiments

**Nelder–Mead algorithm** It is well-known that the Nelder–Mead method efficiently works in low dimensional problems. Indeed, in our preliminary experiments for two-dimensional



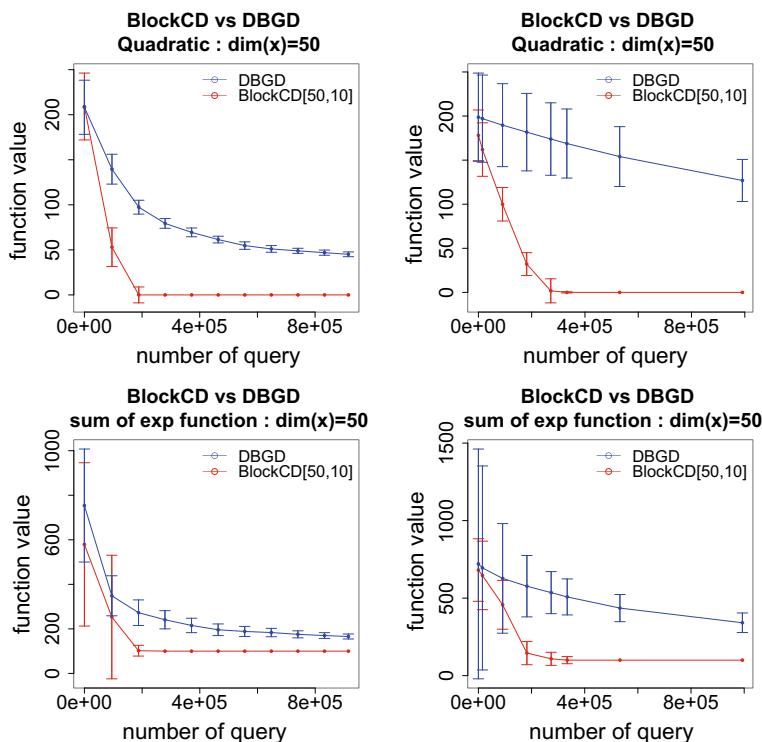
**Fig. 2** *Left panel* two-dimensional quadratic function. *Mid panel* two-dimensional Rosenbrock function. The Nelder–Mead method, BlockCD[2, 1], and BlockCD[2, 2] are compared. For each algorithm, the median of the function value is depicted to the CPU time (s). The vertical bar shows the percentile 30–70 %. *Right panel* The average behavior of the gradient norm of two-dimensional Rosenbrock function in stochastic PC-oracle setting. Although Rosenbrock function is a non-convex function, we can see that the BlockCD algorithm practically converges to a stationary point

optimization problems, the Nelder–Mead method showed a good convergence property compared to the other methods such as BlockCD[2,  $m$ ] with  $m = 1, 2$  (see Fig. 2). We tested optimization methods on the quadratic function  $f(x) = x^\top Ax$  with a random positive definite matrix  $A$ , and two-dimensional Rosenbrock function,  $f(x) = (1 - x_1)^2 + 100(x_2 - x_1^2)^2$ . Rosenbrock function is a famous non-convex test function. The right panel in Fig. 2 shows the average behavior of the true gradient norm of two-dimensional Rosenbrock function in stochastic PC-oracle setting. We randomly generate initial points from normal distribution with mean 0 and variance 2, and conduct 30-trials. We can see that in practice, our proposal, BlockCD algorithm, can achieve the first-order necessary condition about local minimum even when the objective function is non-convex.

**Dueling bandit gradient descent** The dueling bandit gradient descent (DBGD) is a simple online optimization algorithm [21]. In each round, the DBGD firstly sample a unit vector  $u_t$  uniformly. Then, a duel (or a pairwise comparison) between current solution  $x_t$  and a feasible solution  $x'_t = x_t + \delta u_t$  is performed. If  $x_t$  wins then the solution is not updated, otherwise the solution is updated along the direction  $u_t$  by step size  $\gamma$ . In [21], a duel is performed stochastic way. Specifically,  $x_t$  wins  $x'_t$  by a probability

$$\Pr(x_t > x'_t) = \ell(f(x_t) - f(x'_t))$$

where  $f$  is the objective function and  $\ell(\cdot)$  is a link function, e.g. logistic sigmoid function. The largest difference between our proposal and the DBGD is the guarantee of convergence. As we showed in theoretical section, our proposal is guaranteed to averagely converge to the optimal objective function value. On the other hand, the DBGD is only guaranteed sublinear regret which is measured by probability defined above. The algorithmic real performance greatly reflects this difference (see Fig. 3). The experiments are done with test functions described in [21]. The first one is quadratic function  $v_1(x) = x^\top x$  and the second one is the hyperbolic cosine  $v_4(x) = \sum_{i=1}^d \{\exp(x_i) + \exp(-x_i)\}$  (The notation  $v_1$  and  $v_4$  are same as [21]). In this experiment, the performance of two algorithms are evaluated by the number of queries requested in the algorithm. To perform a fair comparison, (i) we use the stochastic PC oracle (2) with repeated querying subroutine (Algorithm 3) in both algorithms, (ii) we set  $\eta$  (tolerance parameter of line search in the BlockCD algorithm) and  $\gamma$  (step size parameter



**Fig. 3** Comparison between the BlockCD and the DBGD with test functions described in [21]. *Top two panels* 50-dimensional quadratic function ( $v_1$  in § 5.1 of [21]). *Bottom two panels* summation of 50-dimensional exponential functions ( $v_4$  in § 5.1 of [21]). For each test function, *left panel* shows the results at  $\eta = 10^{-1} = \gamma$ , where  $\eta$  is tolerance parameter of line search in the BlockCD and  $\gamma$  is step size parameter in the DBGD. Similarly, *right panel* shows the results at  $\eta = 10^{-2} = \gamma$ . Although the DBGD is guaranteed sublinear regret, it can not efficiently decrease the objective function value compared to the BlockCD

in the DBGD algorithm) in the same value. Figure 3 plotted averages and variances of ten times of trials. We can see that the objective function values in the DBGD algorithm can not efficiently converge to the optimal value. This is because the probability that the correct PC information returns comes near  $1/2$  and hence the update which achieves decrease of function value becomes hard to occur. On the other hand, our proposal is shown to converge faster than the DBGD.

## 4.2 Numerical experiments of parallel computation

In BlockCD[ $n, m$ ] with  $m \geq 2$ , one can execute the line search algorithm to each axis separately. Hence, the parallel computation is directly available to find each component of the search direction. Below, we investigate the computational efficiency of the parallel implementation of our method.

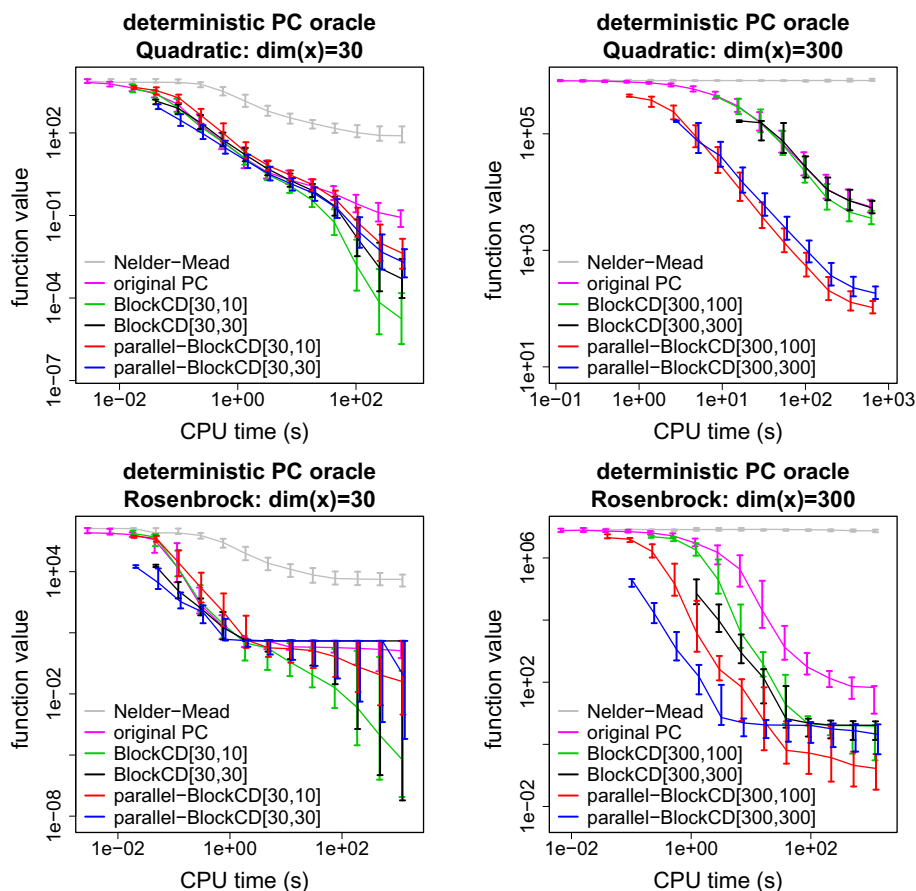
The numerical experiments were conducted on AMD Opteron Processor 6176 (2.3 GHz) with 48 cores (4 processors each of which has 12 cores), running Cent OS Linux release 6.4. In numerical experiments, the R language [17] with snow library for parallel statistical computing was used.

In numerical experiments using the PC oracle, BlockCD[ $n, m$ ] with  $m \geq 2$  and its parallel implementations were compared with the Nelder–Mead algorithm, DBGD, and original PC algorithm, i.e., BlockCD[ $n, 1$ ]. In each iteration of BlockCD[ $n, m$ ] with  $m \geq 2$ ,  $m + 1$  runs of the line search were required. In the parallel implementation, line-search tasks except the search step in Algorithm 1 were almost equally assigned to each processor. In the following, the parallel implementation of BlockCD[ $n, m$ ] is referred to as parallel-BlockCD[ $n, m$ ]. When  $c$  processors are used in parallel-BlockCD[ $n, m$ ] such that  $n$  and  $m$  are much greater than  $c$ , the parallel computation will be approximately  $(m + 1)/(m/c + 1) \approx c$  times more efficient than the serial processing. Practically, however, the communication overhead among processors may cancel the effect of the parallel computation, especially in small-scale problems.

We examined optimization methods on two  $n$ -dimensional optimization problems: the quadratic function  $f(x) = x^\top Ax$ , and Rosenbrock function,  $f(x) = \sum_{i=1}^{n-1} [(1 - x_i)^2 + 100(x_{i+1} - x_i^2)^2]$ . The  $n$  by  $n$  matrix  $A$  was defined as  $A = B^\top B$ , and each element of the  $n$  by  $n$  matrix  $B$  was randomly generated from the standard normal distribution. The quadratic function satisfies the assumptions in Theorem 1. The Rosenbrock function is not convex, and thus, the assumption is violated. We examine whether the proposed method efficiently works even when the theoretical assumptions are not necessarily satisfied. In each objective function, the dimension was set to  $n = 30$  or  $300$ . In all problems, the minimum value is zero. For each algorithm, the optimization was repeated 10 times using randomly chosen initial points generated from  $N(0, 3^2)$ . According to [10], we examined some tuning parameters for the Nelder–Mead algorithm, and we verified that initial points did not significantly affect the numerical results in the present experiments. Hence, the standard parameter setting of the Nelder–Mead algorithm recommended in [10] was used throughout the present experiments.

Numerical results using the deterministic PC oracle are presented in Fig. 4. For each algorithm, the median of function values in optimization process is depicted as the solid line with 30 and 70 percentiles for each CPU time. The Nelder–Mead algorithm did not efficiently work even for 30-dimensional quadratic functions. The performance of the Nelder–Mead algorithm tends to be easily degraded in high dimensional problems, as reported by several authors; see [10] and references therein. The present numerical results agree with past results. The original PC algorithm and serially executed BlockCD[ $n, m$ ] were comparable. As shown in (9), the upper bound of the query complexity is independent of  $m$ , when the deterministic PC oracle is used. As for the efficiency of the parallel computation, the parallel-BlockCD[ $n, m$ ] outperformed the competitors in 300 dimensional problems. In our experiments, the parallel implementation was about 15 times more efficient than the serial implementation in CPU time. For large-scale problems, the communication overhead can be canceled by the efficiency of the parallel computation. In our approach, the parallel computation is easily conducted without losing the convergence property as proved in Theorem 1. Computational efficiency of the parallel computation of our method was verified in moderate-scale optimization problems.

Also, we conducted optimization using the stochastic PC oracle. The DBGD was employed instead of the Nelder–Mead algorithm, since the Nelder–Mead algorithm even with the deterministic PC oracle did not work efficiently. The parameter in the stochastic PC oracle was set to  $\kappa = 2$ ,  $\delta_0 = 0.3$  and  $\mu = 0.01$ . Hence, the probability that the oracle returns the correct sign depends on the difference of function values. According to Lemma 1, queries were repeated at each point so that the probability of receiving the correct sign exceeded  $1 - \delta$ , where  $\delta = 0.1$ . The step-size parameter  $\eta$  in the BlockCD and DBGD was set to

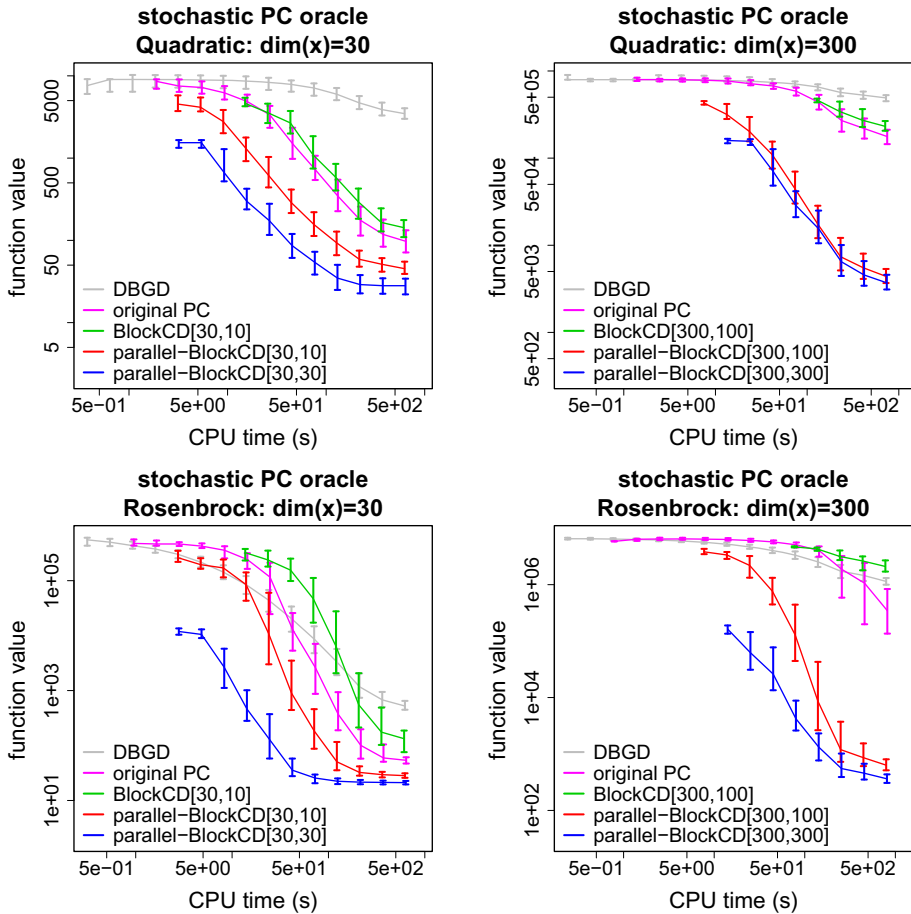


**Fig. 4** Deterministic PC oracle is used in optimization algorithms. *Top panels* results in optimization of quadratic function. *Bottom panels* results in optimization of Rosenbrock function. Nelder-Mead algorithm, original PC algorithm, BlockCD[ $n, m$ ] with  $m = n$  and  $m = n/3$ , and parallel-BlockCD[ $n, m$ ] with  $m = n$  and  $m = n/3$ , are compared for  $n = 30$  and  $n = 300$ . Median of function values is presented to the CPU time (s). The vertical bar shows the percentile 30–70 %

$\eta = 0.01$ . Numerical results are presented in Fig. 5. We dropped results of BlockCD[ $n, n$ ], since BlockCD[ $n, m$ ] with a large  $m$  was extremely inefficient. The convergence rate of the DBGD and original PC algorithm was relatively slow, though the computational cost of each iteration was not high. When the stochastic PC oracle was used, the parallel implementation of BlockCD[ $n, m$ ] achieved a faster convergence rate compared to the other algorithms in CPU time.

## 5 Conclusion

In this paper, we proposed a block coordinate descent algorithm for unconstrained optimization problems using the pairwise comparison of function values. Our algorithm consists of



**Fig. 5** Stochastic PC oracle is used in optimization algorithms. *Top panels* results in optimization of quadratic function. *Bottom panels* results in optimization of Rosenbrock function. DBGD method, original PC algorithm, BlockCD[ $n, m$ ] with  $m = n/3$  and parallel-BlockCD[ $n, m$ ] with  $m = n$  and  $m = n/3$  are compared for  $n = 30$  and  $n = 300$ . Median of function values is shown to the CPU time (s). The vertical bar stands for the percentile 30–70 %

two steps: the direction estimate step and search step. The direction estimate step can easily be parallelized. Hence, our algorithm is effectively applicable to large-scale optimization problems. Theoretically, we obtained an upper bound of the convergence rate and the query complexity, when the deterministic and stochastic pairwise comparison oracles were used. Our upper bound achieves the optimal rate with respect to the query complexity when oracle parameter  $\kappa > 1$ . On the other hand, in practice, our algorithm is simple and easy to implement. In addition, numerical experiments show that the parallel implementation of our algorithm outperformed the other methods using pairwise comparison. An extension of our algorithm to constrained optimization problems is an important future work. Other interesting research directions include pursuing the relation between pairwise comparison oracle and other kind of oracles such as gradient-sign oracle [18].

**Acknowledgments** This work was supported by JSPS KAKENHI Grant No. 16K00044.



## Appendix 1: Proof of Theorem 1

*Proof* The optimal solution of  $f$  is denoted as  $\mathbf{x}^*$ . Let us define  $\varepsilon'$  be  $\varepsilon/(1 + \frac{n}{m\gamma})$ . If  $f(\mathbf{x}_t) - f(\mathbf{x}^*) < \varepsilon'$  holds in the algorithm, we obtain  $f(\mathbf{x}_{t+1}) - f(\mathbf{x}^*) < \varepsilon'$ , since the function value is non-increasing in each iteration of the algorithm

Next, we assume  $\varepsilon' \leq f(\mathbf{x}_t) - f(\mathbf{x}^*)$ . In the following, we use the inequality

$$f(\mathbf{x}_t + \beta_t \mathbf{d}_t / \|\mathbf{d}_t\|) \leq f(\mathbf{x}_t) - \frac{|\nabla f(\mathbf{x}_t)^\top \mathbf{d}_t|^2}{2L\|\mathbf{d}_t\|^2} + \frac{L}{2}\eta^2$$

that is proved in [11]. For the  $i$ th coordinate, let us define the functions  $g_{\text{low},i}(\alpha)$  and  $g_{\text{up},i}(\alpha)$  as

$$g_{\text{low},i}(\alpha) = f(\mathbf{x}_t) + \frac{\partial f(\mathbf{x}_t)}{\partial x_i} \alpha + \frac{\sigma}{2} \alpha^2, \quad \text{and} \quad g_{\text{up},i}(\alpha) = f(\mathbf{x}_t) + \frac{\partial f(\mathbf{x}_t)}{\partial x_i} \alpha + \frac{L}{2} \alpha^2.$$

Then, we have

$$g_{\text{low},i}(\alpha) \leq f(\mathbf{x}_t + \alpha \mathbf{e}_i) \leq g_{\text{up},i}(\alpha).$$

Let  $\alpha_{\text{up},i}$  and  $\alpha_i^*$  be the minimum solution of  $\min_{\alpha} g_{\text{up},i}(\alpha)$  and  $\min_{\alpha} f(\mathbf{x}_t + \alpha \mathbf{e}_i)$ , respectively. In particular,  $\alpha_{\text{up},i}$  can be written explicitly. Then, we obtain

$$g_{\text{low},i}(\alpha_i^*) \leq f(\mathbf{x}_t + \alpha_i^* \mathbf{e}_i) \leq f(\mathbf{x}_t + \alpha_{\text{up},i} \mathbf{e}_i) \leq g_{\text{up},i}(\alpha_{\text{up},i}).$$

The inequality  $g_{\text{low},i}(\alpha_i^*) \leq g_{\text{up},i}(\alpha_{\text{up},i})$  and the concrete form of  $\alpha_{\text{up},i}$  yield that  $\alpha_i^*$  lies between  $-c_0 \frac{\partial f(\mathbf{x}_t)}{\partial x_i}$  and  $-c_1 \frac{\partial f(\mathbf{x}_t)}{\partial x_i}$ , where  $c_0$  and  $c_1$  are defined as

$$c_0 = (1 - \sqrt{1 - \sigma/L})/\sigma, \quad c_1 = (1 + \sqrt{1 - \sigma/L})/\sigma.$$

Here,  $0 < c_0 \leq c_1$  holds. Each component of the search direction  $\mathbf{d}_t = (d_1, \dots, d_n) \neq \mathbf{0}$  in Algorithm 1 satisfies  $|d_i - \alpha_i^*| \leq \eta$  if  $i = i_k$  and otherwise  $d_i = 0$ . For  $I = \{i_1, \dots, i_m\} \subset \{1, \dots, n\}$ , let  $\|\mathbf{a}\|_I^2$  of the vector  $\mathbf{a} \in \mathbb{R}^n$  be  $\sum_{i \in I} a_i^2$ .

The vector  $(\alpha_1^*, \dots, \alpha_n^*)$  is denoted as  $\boldsymbol{\alpha}^*$ . Then, the triangle inequality leads to

$$\begin{aligned} \|\mathbf{d}_t\| &\leq \|\boldsymbol{\alpha}^*\|_I + \|\mathbf{d}_t - \boldsymbol{\alpha}^*\|_I \leq c_1 \|\nabla f(\mathbf{x}_t)\|_I + \sqrt{m}\eta, \\ |\nabla f(\mathbf{x}_t)^\top \mathbf{d}_t| &\geq \left| \sum_{i \in I} (\nabla f(\mathbf{x}_t))_i \alpha_i^* \right| - \left| \sum_{i \in I} (\nabla f(\mathbf{x}_t))_i (d_i - \alpha_i^*) \right| \\ &\geq c_0 \|\nabla f(\mathbf{x}_t)\|_I^2 - \sqrt{m}\eta \|\nabla f(\mathbf{x}_t)\|_I. \end{aligned}$$

The assumption  $\varepsilon' \leq f(\mathbf{x}_t) - f(\mathbf{x}^*)$  leads to

$$2\sigma\varepsilon' \leq 2\sigma(f(\mathbf{x}_t) - f(\mathbf{x}^*)) \leq \|\nabla f(\mathbf{x}_t)\|^2,$$

in which the second inequality is derived from (9.9) in [4].

The above inequality and  $1/4L^2 \leq c_0^2$  yield

$$\eta = \sqrt{\frac{\varepsilon'\sigma}{8L^2n}} \leq c_0 \sqrt{\frac{\sigma\varepsilon'}{2n}} \leq c_0 \frac{\|\nabla f(\mathbf{x}_t)\|}{2\sqrt{n}}.$$

Hence, we obtain

$$\begin{aligned} \|\mathbf{d}_t\| &\leq c_1 \|\nabla f(\mathbf{x}_t)\|_I + \frac{c_0}{2} \sqrt{\frac{m}{n}} \|\nabla f(\mathbf{x}_t)\|, \\ |\nabla f(\mathbf{x}_t)^\top \mathbf{d}_t| &\geq \left[ c_0 \|\nabla f(\mathbf{x}_t)\|_I^2 - \frac{c_0}{2} \sqrt{\frac{m}{n}} \|\nabla f(\mathbf{x}_t)\| \|\nabla f(\mathbf{x}_t)\|_I \right]_+, \end{aligned}$$

where  $[x]_+ = \max\{0, x\}$  for  $x \in \mathbb{R}$ . Let  $Z = \sqrt{\frac{n}{m}} \|\nabla f(\mathbf{x}_t)\|_I / \|\nabla f(\mathbf{x}_t)\|$  be a non-negative valued random variable defined from the random set  $I$ , and define the non-negative value  $k$  as  $k = c_0/c_1 \leq 1$ . A lower bound of the expectation of  $(\|\nabla f(\mathbf{x}_t)\|^\top \mathbf{d}_t / \|\mathbf{d}_t\|)^2$  with respect to the distribution of  $I$  is given as

$$\begin{aligned} \mathbb{E}_I \left[ \left( \frac{|\nabla f(\mathbf{x}_t)^\top \mathbf{d}_t|}{\|\mathbf{d}_t\|} \right)^2 \right] &\geq \mathbb{E}_I \left[ \left( \frac{\left[ c_0 \|\nabla f(\mathbf{x}_t)\|_I^2 - \frac{c_0}{2} \sqrt{\frac{m}{n}} \|\nabla f(\mathbf{x}_t)\| \|\nabla f(\mathbf{x}_t)\|_I \right]_+}{c_1 \|\nabla f(\mathbf{x}_t)\|_I + \frac{c_0}{2} \sqrt{\frac{m}{n}} \|\nabla f(\mathbf{x}_t)\|} \right)^2 \right] \\ &= k^2 \frac{m}{n} \|\nabla f(\mathbf{x}_t)\|^2 \mathbb{E}_I \left[ \frac{Z^2 [Z - 1/2]_+^2}{(Z + k/2)^2} \right] \\ &\geq k^2 \frac{m}{n} \|\nabla f(\mathbf{x}_t)\|^2 \mathbb{E}_I \left[ \frac{Z^2 [Z - 1/2]_+^2}{(Z + 1/2)^2} \right]. \end{aligned}$$

Here, we recall that  $i_k \in I$  is uniformly distributed. The random variable  $Z$  is non-negative, and  $\mathbb{E}_I[Z^2] = 1$  holds. Thus, Lemma 2 below leads to

$$\mathbb{E}_I \left[ \left( \frac{|\nabla f(\mathbf{x}_t)^\top \mathbf{d}_t|}{\|\mathbf{d}_t\|} \right)^2 \right] \geq \frac{k^2}{52} \frac{m}{n} \|\nabla f(\mathbf{x}_t)\|^2.$$

Combining the above inequality with the case of  $f(\mathbf{x}_t) - f(\mathbf{x}^*) < \varepsilon'$ , we obtain the conditional expectation of  $f(\mathbf{x}_{t+1}) - f(\mathbf{x}^*)$  for given  $\mathbf{d}_0, \mathbf{d}_1, \dots, \mathbf{d}_{t-1}$  as follows.

$$\begin{aligned} &\mathbb{E} [f(\mathbf{x}_{t+1}) - f(\mathbf{x}^*) | \mathbf{d}_0, \dots, \mathbf{d}_{t-1}] \\ &\leq \mathbf{1} [f(\mathbf{x}_t) - f(\mathbf{x}^*) \geq \varepsilon'] \cdot \left[ f(\mathbf{x}_t) - f(\mathbf{x}^*) - \frac{k^2}{104L} \frac{m}{n} \|\nabla f(\mathbf{x}_t)\|^2 + \frac{L\eta^2}{2} \right] \\ &\quad + \mathbf{1} [f(\mathbf{x}_t) - f(\mathbf{x}^*) < \varepsilon'] \cdot \varepsilon' \\ &\leq \mathbf{1} [f(\mathbf{x}_t) - f(\mathbf{x}^*) \geq \varepsilon'] \cdot \left[ \left( 1 - \frac{m}{n} \gamma \right) (f(\mathbf{x}_t) - f(\mathbf{x}^*)) + \frac{L\eta^2}{2} \right] \\ &\quad + \mathbf{1} [f(\mathbf{x}_t) - f(\mathbf{x}^*) < \varepsilon'] \cdot \varepsilon'. \end{aligned} \tag{18}$$

Taking the expectation with respect to all  $\mathbf{d}_0, \dots, \mathbf{d}_t$  yields

$$\begin{aligned} \mathbb{E} [f(\mathbf{x}_{t+1}) - f(\mathbf{x}^*)] &\leq \left( 1 - \frac{m}{n} \gamma \right) \mathbb{E} [\mathbf{1} [f(\mathbf{x}_t) - f(\mathbf{x}^*) \geq \varepsilon'] (f(\mathbf{x}_t) - f(\mathbf{x}^*))] \\ &\quad + \mathbb{E} [\mathbf{1} [f(\mathbf{x}_t) - f(\mathbf{x}^*) \geq \varepsilon']] \frac{L\eta^2}{2} + \mathbb{E} [\mathbf{1} [f(\mathbf{x}_t) - f(\mathbf{x}^*) < \varepsilon']] \varepsilon' \\ &\leq \left( 1 - \frac{m}{n} \gamma \right) \mathbb{E} [f(\mathbf{x}_t) - f(\mathbf{x}^*)] + \max \left\{ \frac{L\eta^2}{2}, \varepsilon' \right\}. \end{aligned}$$

Since  $0 < \gamma < 1$  and  $\max\{L\eta^2/2, \varepsilon'\} = \varepsilon'$  hold, for  $\Delta_T = \mathbb{E}[f(\mathbf{x}_T) - f(\mathbf{x}^*)]$  we have

$$\Delta_T - \frac{n}{m} \frac{\varepsilon'}{\gamma} \leq \left( 1 - \frac{m}{n} \gamma \right) \left( \Delta_{T-1} - \frac{n}{m} \frac{\varepsilon'}{\gamma} \right) \leq \left( 1 - \frac{m}{n} \gamma \right)^T \Delta_0.$$

When  $T$  is greater than  $T_0$  in (6), we obtain  $(1 - \frac{m}{n} \gamma)^T \Delta_0 \leq \varepsilon'$  and

$$\Delta_T \leq \varepsilon' \left( 1 + \frac{n}{m\gamma} \right) = \varepsilon.$$

□

**Remark 7** For  $m = 1$ , the exact evaluation of  $\mathbb{E}[(\nabla f(\mathbf{x}_t)^\top \mathbf{d}_t / \|\mathbf{d}_t\|)^2]$  is possible. Hence, we do not need to introduce the threshold  $\varepsilon'$  to evaluate the perturbation of the norm  $\|\mathbf{d}_t\|$  such as (18). Arbitrary small  $\varepsilon'$  is available, and  $\max\{L\eta^2/2, \varepsilon'\}$  in the above proof becomes  $L\eta^2/2$ . As a result, the faster convergence rate shown in [11] is obtained for  $m = 1$ .

**Lemma 2** Let  $Z$  be a non-negative random variable satisfying  $\mathbb{E}[Z^2] = 1$ . Then, we have

$$\mathbb{E}\left[Z^2 \frac{[Z - 1/2]_+^2}{(Z + 1/2)^2}\right] \geq \frac{1}{52}.$$

*Proof* For  $z \geq 0$  and  $\delta \geq 0$ , we have the inequality

$$\frac{[z - 1/2]_+^2}{(z + 1/2)^2} \geq \frac{\delta^2}{(1 + \delta)^2} \mathbf{1}[z \geq 1/2 + \delta].$$

Then, we get

$$\begin{aligned} \mathbb{E}\left[Z^2 \frac{[Z - 1/2]_+^2}{(Z + 1/2)^2}\right] &\geq \frac{\delta^2}{(1 + \delta)^2} \mathbb{E}[Z^2 \mathbf{1}[Z \geq 1/2 + \delta]] \\ &= \frac{\delta^2}{(1 + \delta)^2} \mathbb{E}[Z^2 (1 - \mathbf{1}[Z < 1/2 + \delta])] \\ &= \frac{\delta^2}{(1 + \delta)^2} (1 - \mathbb{E}[Z^2 \mathbf{1}[Z < 1/2 + \delta]]) \\ &\geq \frac{\delta^2}{(1 + \delta)^2} (1 - (1/2 + \delta)^2 \Pr(Z < 1/2 + \delta)) \\ &\geq \frac{\delta^2}{(1 + \delta)^2} (1 - (1/2 + \delta)^2). \end{aligned}$$

By setting  $\delta$  appropriately, we obtain

$$\mathbb{E}_I\left[Z^2 \frac{[Z - 1/2]_+^2}{(Z + 1/2)^2}\right] \geq \frac{1}{52}.$$

□

## Appendix 2: Proof of Corollary 1

*Proof* Let us replace  $T_0$  and  $K_0$  with

$$T_0 = \frac{n}{m\gamma} \log \frac{\Delta_0(1 + \frac{n}{m\gamma})}{\varepsilon} + 1 = \frac{n}{m\gamma} \log \frac{e^{m\gamma/n} \Delta_0(1 + \frac{n}{m\gamma})}{\varepsilon}, \quad (19)$$

$$K_0 = \frac{2}{\log 2} \log \frac{2^{13}(L/\sigma)^3 n \Delta_0(1 + \frac{n}{m\gamma})}{\varepsilon} + 1 = \frac{2}{\log 2} \log \frac{2^{13.5}(L/\sigma)^3 n \Delta_0(1 + \frac{n}{m\gamma})}{\varepsilon}, \quad (20)$$

respectively. Note that  $e^{m\gamma/n} \Delta_0(1 + \frac{n}{m\gamma}) \leq 2^{13.5}(L/\sigma)^3 n \Delta_0(1 + \frac{n}{m\gamma})$  holds because of  $\sigma \leq L$ ,  $\gamma < 1$  and  $1 \leq m \leq n$ . Suppose that

$$2^{13.5}(L/\sigma)^3 n \Delta_0 \left(1 + \frac{n}{m\gamma}\right) \leq \frac{1}{\varepsilon} \quad (21)$$

holds. Then we have

$$T_0 \leq \frac{2n}{m\gamma} \log \frac{1}{\varepsilon}, \quad K_0 \leq \frac{4}{\log 2} \log \frac{1}{\varepsilon},$$

and thus,

$$(m+2)T_0K_0 \leq \frac{8}{\log 2} \frac{(m+2)n}{m\gamma} \left( \log \frac{1}{\varepsilon} \right)^2 =: Q_0$$

holds. Theorem 1 leads to

$$\mathbb{E} [f(\mathbf{x}_{Q_0}) - f(\mathbf{x}^*)] \leq \varepsilon = \exp \left\{ -c_0 \sqrt{\frac{m\gamma}{m+2}} \times \frac{Q_0}{n} \right\},$$

where  $c_0 = \sqrt{\frac{\log 2}{8}}$ . The condition (21) is expressed as

$$\frac{n}{c_0^2} \frac{m+2}{m\gamma} \left[ \log \left( 2^{13.5} \left( \frac{L}{\sigma} \right)^3 n \Delta_0 \left( 1 + \frac{n}{m\gamma} \right) \right) \right]_+^2 \leq Q_0,$$

where  $[a]_+ = \max\{a, 0\}$ . Note that the left-hand side of the above inequality is determined from the problem setup (i.e.  $\sigma$ ,  $L$ ,  $n$ ), initial point  $\mathbf{x}_0$ , and the parallelization parameter  $m$  of Algorithm 1.  $\square$

### Appendix 3: Proof of Corollary 2

*Proof* For the output  $\hat{\mathbf{x}}_Q$  of BlockCD[ $n, m$ ],  $f(\hat{\mathbf{x}}_Q) \geq f(\hat{\mathbf{x}}_{Q+1})$  holds, and thus, the sequence  $\{\hat{\mathbf{x}}_Q\}_{Q \in \mathbb{N}}$  is included in

$$C(\mathbf{x}_0) := \{\mathbf{x} \in \mathbb{R}^n \mid f(\mathbf{x}) \leq f(\mathbf{x}_0)\}.$$

Since  $f$  is convex and continuous,  $C(\mathbf{x}_0)$  is convex and closed. Moreover, since  $f$  is convex and it has non-degenerate Hessian, the Hessian is positive definite, and thus,  $f$  is strictly convex. Then  $C(\mathbf{x}_0)$  is bounded as follows. We set

the minimal directional derivative along the radial direction from  $\mathbf{x}^*$  over the unit sphere around  $\mathbf{x}^*$  as

$$b := \min_{\|u\|=1} \nabla f(\mathbf{x}^* + u) \cdot u.$$

Then,  $b$  is strictly positive and the following holds for any  $\mathbf{x} \in C(\mathbf{x}_0)$  such that  $\|\mathbf{x} - \mathbf{x}^*\| \geq 1$ ,

$$b\|\mathbf{x} - \mathbf{x}^*\| + (f(\mathbf{x}^*) - b) \leq f(\mathbf{x}) \leq f(\mathbf{x}_0).$$

Thus we have

$$C(\mathbf{x}_0) \subset \left\{ \mathbf{x} \mid \|\mathbf{x} - \mathbf{x}^*\| \leq 1 + \frac{f(\mathbf{x}_0) - f(\mathbf{x}^*)}{b} \right\}. \quad (22)$$

Since the right hand side of (22) is a bounded ball,  $C(\mathbf{x}_0)$  is also bounded.

Thus,  $C(\mathbf{x}_0)$  is a convex compact set.

Since  $f$  is twice continuously differentiable, the Hessian matrix  $\nabla^2 f(\mathbf{x})$  is continuous with respect to  $\mathbf{x} \in \mathbb{R}^n$ . By the positive definiteness of the Hessian matrix, the minimum and maximum eigenvalues  $e_{\min}(\mathbf{x})$  and  $e_{\max}(\mathbf{x})$  of  $\nabla^2 f(\mathbf{x})$  are continuous and positive.

Therefore, there are the positive minimum value  $\sigma$  of  $e_{\min}(\mathbf{x})$  and maximum value  $L$  of  $e_{\max}(\mathbf{x})$  on the compact set  $C(\mathbf{x}_0)$ . It means that  $f$  is  $\sigma$ -strongly convex and  $L$ -Lipschitz on  $C(\mathbf{x}_0)$ . Thus, the same argument to obtain (9) can be applied for  $f$ .  $\square$

## References

1. Audet, C., Dennis Jr., J.E.: Analysis of generalized pattern searches. *SIAM J. Optim.* **13**(3), 889–903 (2002)
2. Audet, C., Dennis Jr., J.E., Digabel, S.L.: Parallel space decomposition of the mesh adaptive direct search algorithm. *SIAM J. Optim.* **19**(3), 1150–1170 (2008)
3. Audet, C., Ianni, A., Le Digabel, S., Tribes, C.: Reducing the number of function evaluations in mesh adaptive direct search algorithms. *SIAM J. Optim.* **24**(2), 621–642 (2014)
4. Boyd, S.P., Vandenberghe, L.: *Convex Optimization*. Cambridge University Press, Cambridge (2004)
5. Conn, A.A.R., Scheinberg, K., Vicente, L.N.: *Introduction to Derivative-Free Optimization*, vol. 8. SIAM, Philadelphia (2009)
6. Conn, A.R., Le Digabel, S.: Use of quadratic models with mesh-adaptive direct search for constrained black box optimization. *Optim. Methods Softw.* **28**(1), 139–158 (2013)
7. Custódio, A., Dennis, J., Vicente, L.N.: Using simplex gradients of nonsmooth functions in direct search methods. *IMA J. Numer. Anal.* **28**(4), 770–784 (2008)
8. Flaxman, A.D., Kalai, A.T., McMahan, H.B.: Online convex optimization in the bandit setting: gradient descent without a gradient. In: *Proceedings of the sixteenth annual ACM-SIAM symposium on discrete algorithms. SODA '05*, pp. 385–394. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA (2005)
9. Fu, M.C.: Gradient estimation. In: Henderson, S.G., Nelson, B.L. (eds.) *Handbooks in Operations Research and Management Science: Simulation*, Chap. 19. Elsevier, Amsterdam (2006)
10. Gao, F., Han, L.: Implementing the Nelder–Mead simplex algorithm with adaptive parameters. *Comput. Optim. Appl.* **51**(1), 259–277 (2012)
11. Jamieson, K.G., Nowak, R.D., Recht, B.: Query complexity of derivative-free optimization. In: *NIPS*, pp. 2681–2689 (2012)
12. Kääriäinen, M.: Active learning in the non-realizable case. In: *Algorithmic Learning Theory*, pp. 63–77. Springer (2006)
13. Lagarias, J.C., Reeds, J.A., Wright, M.H., Wright, P.E.: Convergence properties of the Nelder–Mead simplex method in low dimensions. *SIAM J. Optim.* **9**, 112–147 (1998)
14. Luenberger, D., Ye, Y.: *Linear and Nonlinear Programming*. Springer, Berlin (2008)
15. Mohri, M., Rostamizadeh, A., Talwalkar, A.: *Foundations of Machine Learning*. MIT Press, London (2012)
16. Nelder, J.A., Mead, R.: A simplex method for function minimization. *Comput. J.* **7**(4), 308–313 (1965)
17. R Core Team: *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria (2014). <http://www.R-project.org/>
18. Ramdas, A., Singh, A.: Algorithmic connections between active learning and stochastic convex optimization. In: *Algorithmic Learning Theory*, pp. 339–353. Springer (2013)
19. Richtárik, P., Takáč, M.: Parallel coordinate descent methods for big data optimization. *Math. Program.* **156**(1–2), 433–484 (2016)
20. Rios, L.M., Sahinidis, N.V.: Derivative-free optimization: a review of algorithms and comparison of software implementations. *J. Glob. Optim.* **56**(3), 1247–1293 (2013)
21. Yue, Y., Joachims, T.: Interactively optimizing information retrieval systems as a dueling bandits problem. In: *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 1201–1208. ACM (2009)
22. Yue, Y., Broder, J., Kleinberg, R., Joachims, T.: The k-armed dueling bandits problem. *J. Comput. Syst. Sci.* **78**(5), 1538–1556 (2012)
23. Zoghi, M., Whiteson, S., Munos, R., de Rijke, M.: Relative upper confidence bound for the K-armed dueling bandit problem. In: *ICML 2014: Proceedings of the Thirty-First International Conference on Machine Learning*, pp. 10–18 (2014)