Incremental Recursive Ranking Grouping for Large Scale Global Optimization

Marcin Michal Komarnicki, Michal Witold Przewozniczek, Halina Kwasnicka, and Krzysztof Walkowiak

***Abstract*—Real-world optimization problems may have a different underlying structure. In black-box optimization, the dependencies between decision variables remain unknown. However, some techniques can discover such interactions accurately. In Large Scale Global Optimization (LSGO), problems are high-dimensional. It was shown effective to decompose LSGO problems into subproblems and optimize them separately. The effectiveness of such approaches may be highly dependent on the accuracy of problem decomposition. Many state-of-the-art decomposition strategies are derived from Differential Grouping (DG). However, if a given problem consists of non-additively separable subproblems, DG-based strategies may discover many non-existing interactions. On the other hand, monotonicity checking strategies proposed so far do not report non-existing interactions for any separable subproblems but may miss discovering many of the existing ones. Therefore, we propose Incremental Recursive Ranking Grouping (IRRG) that suffers from none of these flaws. IRRG consumes more fitness function evaluations than the recent DG-based propositions, e.g., Recursive DG 3 (RDG3). Nevertheless, the effectiveness of the considered Cooperative Co-evolution frameworks after embedding IRRG or RDG3 was similar for problems with additively separable subproblems that are suitable for RDG3. After replacing the additive separability with non-additive, embedding IRRG leads to results of significantly higher quality.**

***Index Terms*—Large scale global optimization, problem decomposition, monotonicity checking, non-additive separability.**

I. Introduction

In practice, many continuous Large Scale Global Optimization (LSGO) problems emerge [1]–[3]. They were classified as *complex continuous optimization problems* [4]. Thus, solving them may be considered important but difficult. Real-value encoded instances are classified as large if they have at least 500 decision variables. However, it is frequent to consider the LSGO instances with at least 1000 variables [1], [4]. The size of the search space is exponentially proportional to the number of problem dimensions [4]. Thus, increasing the dimensionality may cause the exponential growth of the number of local optima [1] or the enormously high cost of running Estimation of Distribution Algorithms [5]. Therefore, tackling LSGO instances may be hard for Evolutionary Algorithms designed to solve less dimensional optimization problems.

In the state-of-the-art LSGO-dedicated optimizers, two main approaches are considered. The first, involves hybrid optimization methods, which do not utilize problem decomposition, e.g., Success-History Based Parameter Adaptation for Differential Evolution with Iterative Local Search (SHADE-ILS) [6]. The other approach is to decompose a problem first and then optimize its subproblems independently, frequently using Cooperative Co-evolution (CC) [7]–[10]. Nowadays, the state-of-the-art problem decomposition strategies derive from Differential Grouping (DG) [11]–[16]. These strategies assume that problem consists of additively separable subproblems. If this assumption is false, the DG-based strategies may report *false linkage* that takes place when two independent variables are found dependent [17]. False linkage may significantly decrease the quality of the decomposition and decrease the overall method effectiveness [18]. Decomposition strategies that do not assume the existence of only additively separable subproblems are based on monotonicity checking [19]–[22]. However, they may miss finding some of the interactions (*missing linkage* [17]), and DG-based strategies where shown to be less vulnerable to this inconsistency [11], [13], [14], [22].

In this article, we propose IRRG, a new problem decomposition strategy that does not assume the existence of only additively separable subproblems. IRRG is derived from monotonicity checking, thus, it never reports false linkage. Additionally, it significantly limits the issue of missing linkage. This paper has two main objectives. First, we show that IRRG can decompose continuous LSGO problems accurately regardless of the additive or non-additive separability. Second, we show that although IRRG is more expensive than Recursive DG 3 (RDG3) [16], its influence on the overall optimization cost is negligible also when the problems are DG-suitable. To meet these objectives, we embed both, IRRG and RDG3, into two different CC frameworks. We consider a set of 45 problems including additively and non-additively separable ones. This set is built from the CEC'2013 functions [23]. Since the standard CEC'2013 set contains only one function with non-additively separable subfunctions, we propose two transformations of the additive separability into non-additive.

The rest of this article is organized as follows. The related work is discussed in the next section. In Section III, we compare DG and monotonicity checking. Section IV presents the details of IRRG, whereas the fifth section reports the results of the experiments. Finally, in the last section, we conclude this paper and propose the directions of future work.

II. Related Work

A. Problem Decomposition

The quality of the problem decomposition is related to the separability structure of its objective function [13]. A function

Manuscript received April 22, 2022; revised August 29, 2022. This work was supported by the Polish National Science Centre (NCN) under Grant 2020/38/E/ST6/00370.

The authors are with the Wroclaw University of Science and Technology, 50-370 Wroclaw, Poland (e-mail: marcin.komarnicki@pwr.edu.pl; michal.przewozniczek@pwr.edu.pl; halina.kwasnicka@pwr.edu.pl; krzysztof.walkowiak@pwr.edu.pl).

$f : \Omega \to \mathbb{R}$ is partially separable [24], i.e., consists of m independent subfunctions, $2 \leq m \leq n$, if

$$\arg\min_{\mathbf{x}} f(\mathbf{x}) = \left[\arg\min_{\mathbf{x_1}} f(\mathbf{x_1}, ...), ..., \arg\min_{\mathbf{x_m}} f(..., \mathbf{x_m}) \right] \quad (1)$$

where $\mathbf{x} \in \Omega$ is a vector of n decision variables $[x_1, ..., x_n]$, $\forall_{i \in \{1,...,m\}} \mathbf{x_i} \in \Omega_i$, and $\Omega = \Omega_1 \times ... \times \Omega_m$. When $m = n$ then each decision variable does not interact with the others and a function is fully separable.

Problem decomposition strategies include uniformed decomposition [25], random grouping [26], delta grouping [27], meta modelling [28], formula-based grouping [29], differential grouping [11]–[16], and monotonicity checking [19]–[22]. The last two approaches are highly related to this paper's scope. Thus, we describe them in detail.

1) Differential Grouping: Many real-world optimization problems are partially separable [30], e.g., additively separable [11]. The definition of an additively separable function is similar to formula (1): $f(\mathbf{x}) = \sum_{i=1}^{m} f_i(\mathbf{x_i})$. Differential Grouping (DG) [11] is a problem decomposition strategy proven to group only dependent variables if a problem is additively separable. DG marks two variables as interacting if $\exists a, b_1 \neq b_2, \delta > 0, \mathbf{x}^* \in \Omega$ the following condition holds

$$\Delta_{\delta,x_p}[f](\mathbf{x}^*)|_{x_p=a,x_q=b_1} \neq \Delta_{\delta,x_p}[f](\mathbf{x}^*)|_{x_p=a,x_q=b_2} \quad (2)$$

where x_p and x_q indicate the pth and qth problem variables, whereas the definition of function $\Delta_{\delta,x_p}[f]$ is as follows

$$\Delta_{\delta,x_p}[f](\mathbf{x}) = f(..., x_p + \delta, ...) - f(..., x_p, ...) \quad (3)$$

By $f(\mathbf{x})|_{x_p=a,x_q=b}$ we mean the value of f after setting the values of x_p and x_q to a and b, respectively. Each perturbation of the pth variable, i.e., $x_p + \delta$, must produce a feasible solution. Without loss of generality, we may denote the left and right sides of formula (2) as Δ_1 and Δ_2, respectively. Due to the inaccuracy of the float numbers representation, a user-defined ϵ is employed to control the sensitivity of DG: $|\Delta_1 - \Delta_2| > \epsilon$ instead of $\Delta_1 \neq \Delta_2$. Its appropriate value may be different for various optimization problems [13]. DG optimizes the number of pair-wise checks (the single interaction check requires four fitness function evaluations (FFEs)) by ignoring some pairs of variables. The worst scenario is when problem is fully separable, then, $2n(n-1)$ FFEs are needed. The minimal cost, for a fully non-separable problem, is $4(n-1)$ FFEs. Thus, the DG's time complexity is $\mathcal{O}(n^2)$.

DG2 is an improved version of DG [13]. It decreases by half the FFE cost of decomposition for fully separable problems. Additionally, DG2 can detect the overlapping structure of a given optimization problem by checking all possible pairs of variables. In DG2, ϵ is automatically computed as follows. Each interaction check requires fitness values of four solutions: $\mathbf{x_1^*}$, $\mathbf{x_2^*}$, $\mathbf{x_3^*}$, and $\mathbf{x_4^*}$. Thus, $|\Delta_1 - \Delta_2| > \epsilon$ can be reformulated as $|[f(\mathbf{x_1^*}) - f(\mathbf{x_2^*})] - [f(\mathbf{x_3^*}) - f(\mathbf{x_4^*})]| > \epsilon$, where

$$\epsilon = f_\gamma(\sqrt{n} + 2) \cdot (|f(\mathbf{x_1^*})| + |f(\mathbf{x_2^*})| + |f(\mathbf{x_3^*})| + |f(\mathbf{x_4^*})|) \quad (4)$$

$$f_\gamma(k) = \frac{k\mu_M}{1 - k\mu_M} \quad (5)$$

where μ_M is a machine dependent constant [31].

2) Recursive Differential Grouping: Recursive Differential Grouping (RDG) [14] reduces the complexity of DG2 from $\mathcal{O}(n^2)$ to $\mathcal{O}(n \log(n))$. DG and DG2 perform the pair-wise interaction check, whereas RDG consider two disjoint groups of variables X_1 and X_2 that are subsets of $X = \{x_1, ..., x_n\}$. Groups interact if at least one pair of variables $x_p \in X_1$ and $x_q \in X_2$ is non-separable. Then, there exist such values δ_1, $\delta_2 > 0$, unit vectors $\mathbf{u_1} \in U_{X_1}$, $\mathbf{u_2} \in U_{X_2}$, and decision vector $\mathbf{x}^* \in \Omega$ that meet

$$f(\mathbf{x}^* + \delta_1\mathbf{u_1} + \delta_2\mathbf{u_2}) - f(\mathbf{x}^* + \delta_2\mathbf{u_2}) \neq f(\mathbf{x}^* + \delta_1\mathbf{u_1}) - f(\mathbf{x}^*) \quad (6)$$

where $\mathbf{u_j} = [u_1, ..., u_n] \in U_{X_j}$ such that $\forall_{i \in \{1,...,n\}} u_i = 0 \Leftrightarrow x_i \notin X_j$. Thus, RDG is executed recursively, dividing subsequent groups of decision variables by half until a one-element group is reached or the separability is discovered.

RDG3 [16], introduces two new thresholds, i.e., ϵ_s and ϵ_n. It tries to join all separable variables into groups of size ϵ_s. The ϵ_n threshold is useful for overlapping problems. The variables of such problems are divided into groups of approximately ϵ_n size instead of creating one group. CC frameworks embedding RDG3 were shown more effective than those using its predecessors [16].

3) Monotonicity Checking: Monotonicity checking strategies [19]–[21] do not assume that a problem is additively separable. Two decision variables x_p and x_q are interacting if $\exists a_1 \neq a_2, b_1 \neq b_2, \mathbf{x}^* \in \Omega$ such that

$$\begin{aligned} &f(\mathbf{x}^*)|_{x_p=a_1,x_q=b_1} \leq f(\mathbf{x}^*)|_{x_p=a_2,x_q=b_1} \wedge \\ &f(\mathbf{x}^*)|_{x_p=a_1,x_q=b_2} > f(\mathbf{x}^*)|_{x_p=a_2,x_q=b_2} \end{aligned} \quad (7)$$

Fast variable interdependence learning (FVIL) [22] replaces this pair-wise interaction check by examining two disjoint groups of decision variables X_1 and X_2. They are interacting if $\exists \delta_1, \delta_2 > 0$, $\mathbf{u_1} \in U_{X_1}$, $\mathbf{u_2} \in U_{X_2}$, $\mathbf{x}^* \in \Omega$ such that

$$f(\mathbf{x}^*) \leq f(\mathbf{x}^* + \delta_1\mathbf{u_1}) \wedge f(\mathbf{x}^* + \delta_2\mathbf{u_2}) > f(\mathbf{x}^* + \delta_1\mathbf{u_1} + \delta_2\mathbf{u_2}) \quad (8)$$

To check if X_1 and X_2 interact, randomly created δ_1, δ_2, $\mathbf{u_1}$, $\mathbf{u_2}$, and $\mathbf{x}^*$ are used. This procedure is repeated at most N times. As in RDG, subsequent groups of variables are being recursively divided. Thus, the complexity is also $\mathcal{O}(n \log(n))$.

B. Cooperative Co-Evolution and Hybrid Optimization

The idea behind CC is to optimize each component (subproblem) separately. Thus, the decomposition strategies are useful for CC [7]. The decomposition quality is not the only factor that influences the effectiveness of CC—the lower quality decomposition may lead to better results [16]. In the original CC, the expenses for optimizing each component are similar. Since different subproblems may have a different (high or low) impact on global fitness improvements [8], it seems reasonable to concentrate the optimization on those components that will have the highest impact on fitness.

Contribution-based CC (CBCC) [8], [10] computes an accumulated contribution ΔF_i of the ith component considering all previous fitness improvements. Improvements found earlier have a lower impact on the accumulated contribution. CBCC does not require components of equal sizes. A subpopulation of a different size may optimize each component, but the FFE budget of the single run is the same for each component. In

CCFR2 (another extended CC framework) [9], each subpopulation (that may be of a different size as in CBCC) is executed every time for the same number of iterations. Additionally, in CCFR2, the accumulated contribution considers the computational cost of finding the improvement.

In CBCC and CCFR2, we provide them the decomposition of a considered problem and execute separate optimization processes for separate components. In hybrid methods, we search for the most appropriate optimizer for optimizing all variables together or their random subset [6], [32], [33]. SHADE-ILS [6] hybridizes SHADE [34] and two local search methods. SHADE is used to explore the search space, while local search is used to improve the quality of promising solutions. SHADE-ILS employs MTS-L1 [35] and L-BFGS-B [36] that have complementary pros and conns. MTS-L1 is effective for separable problems, while L-BFGS-B approximates the gradient and is more robust. Additionally, SHADE-ILS restarts itself when the improvement ratio is low.

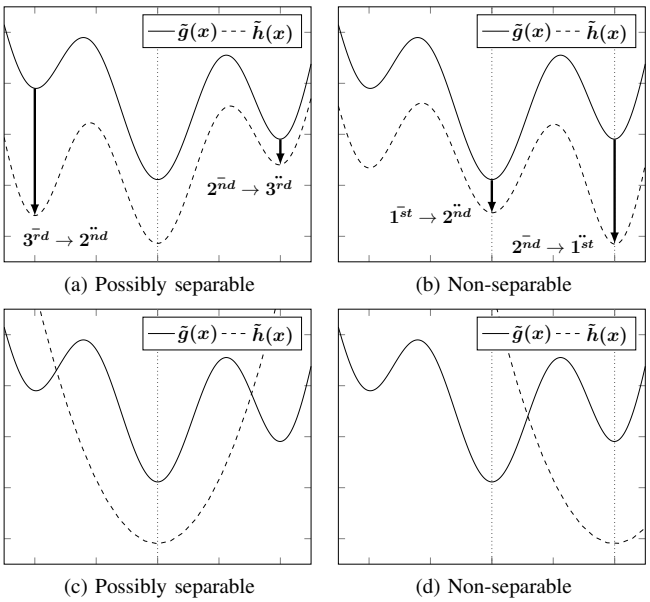
III. Differential Grouping and Monotonicity Checking—Analysis of Main Differences

This section presents a comparison between differential grouping and monotonicity checking. We focus on functions for which at least one of these two strategies reports false or missing linkage. To this end, we adapt the definition of the variable separability from [17] to continuous domains. For the sake of clarity, we consider a two variable function $\tilde{f}$ and two univariate subfunctions that simplify $\tilde{f}$ by replacing the second variable by a constant value: $\tilde{g}(x) = \tilde{f}(\mathbf{x})|_{x_1=x,x_2=b_1}$ and $\tilde{h}(x) = \tilde{f}(\mathbf{x})|_{x_1=x,x_2=b_2}$, where b_1 and b_2 are two different constant values. Conclusions drawn for a two variable function can be easily generalized to more variables.

A. Monotonicity Checking as Empirical Linkage Learning

According to formula (1), and taking subfunctions $\tilde{g}$ and $\tilde{h}$ into account, we can state that if $\arg\min_x \tilde{g}(x) \neq \arg\min_x \tilde{h}(x)$, then x_1 and x_2 are interacting. Otherwise, although the global optimum is not changed, it is not certain that x_1 and x_2 are separable, because only two different values of x_2, i.e., b_1 and b_2, have been considered. After setting x_2 to b_3 and b_4, such that $b_3 \neq b_1 \wedge b_4 \neq b_2$, the condition $\arg\min_x \tilde{g}(x) \neq \arg\min_x \tilde{h}(x)$ could be met. This kind of interaction condition can omit fitness landscape characteristics [37], which may significantly affect the optimizer effectiveness [38], [39]. Fig. 1a and 1b present cases when one of optima becomes better than another. However, only if a global one changes, the variables are found interacting. Similarly, if a subfunction transforms from multi- into unimodal for different values of x_2, as in Fig. 1c and 1d, the dependency will be discovered only when a global optimum is affected.

The provided examples show that even if two variables are possibly separable (according to formula (1)), an optimizer may behave differently depending on which subfunctions will be taken into account. Thus, the condition comparing only global optimum changes to distinguish between the separability and the non-separability seems insufficient. In the example presented in Fig. 2, we consider a greedy optimizer that



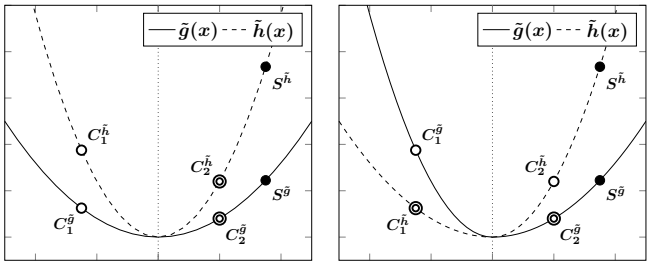
(a) Possibly separable (b) Non-separable

(c) Possibly separable (d) Non-separable

Fig. 1. Separability according to formula (1)

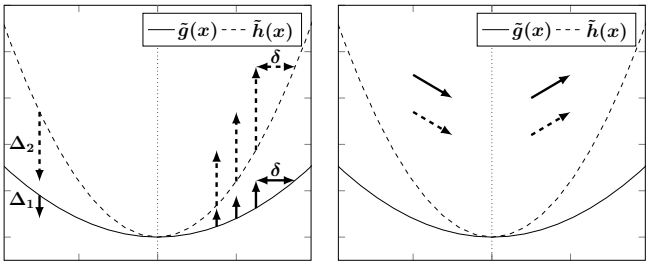
starts its search in point $S^{\tilde{g}}$ or $S^{\tilde{h}}$ for subfunctions $\tilde{g}$ and $\tilde{h}$, respectively. Since $\tilde{g}$ and $\tilde{h}$ have the same global optimum, then x_1 and x_2 are possibly separable according to formula (1). Let us consider the following candidate solutions, $C_1^{\tilde{g}}$, $C_2^{\tilde{g}}$ for subfunction $\tilde{g}$, and $C_1^{\tilde{h}}$, $C_2^{\tilde{h}}$ for subfunction $\tilde{h}$. Double circles indicate the better candidate solution for $\tilde{g}$ and $\tilde{h}$. In Fig. 2a, the choice is the same for $\tilde{g}$ and $\tilde{h}$. Thus, x_1 and x_2 may be, indeed, separable. In Fig. 2b the choice is different for each subfunction. The optimization process of $\tilde{g}$ and $\tilde{h}$ may result in different points in the search space. Therefore, it seems reasonable to state that the variables x_1 and x_2 are interacting. For more complex subfunctions, an optimizer may be deceived to a local optimum only for one subfunction, although both have the same global optimum. The selection pressure favors better fitted individuals. Thus, a single change in the outcome of the individuals' fitness comparison may influence a direction of the search [40].

The above remarks are consistent with the idea of empirical linkage learning (ELL) [17], a new class of linkage learning techniques. In ELL, interactions are discovered by comparing local search results before and after perturbing one of the decision variables. Two variables interact if a local search method converges to a different value after perturbing one of them. Such techniques are proven never to report false linkage. The formal proof assumes the following definition of the variable separability [17]. Two disjoint sets of variables X_1 and X_2 are independent if for each values δ_1, $\delta_2 > 0$, unit vectors $\mathbf{u_1} \in U_{X_1}$, $\mathbf{u_2} \in U_{X_2}$, and decision vector $\mathbf{x}^* \in \Omega$ the following condition holds: $f(\mathbf{x}^*) < f(\mathbf{x}^* + \delta_1\mathbf{u_1}) \Leftrightarrow f(\mathbf{x}^* + \delta_2\mathbf{u_2}) < f(\mathbf{x}^* + \delta_1\mathbf{u_1} + \delta_2\mathbf{u_2})$ and $f(\mathbf{x}^*) = f(\mathbf{x}^* + \delta_1\mathbf{u_1}) \Leftrightarrow f(\mathbf{x}^* + \delta_2\mathbf{u_2}) = f(\mathbf{x}^* + \delta_1\mathbf{u_1} + \delta_2\mathbf{u_2})$ on the premise that all arguments of f are feasible solutions. In the latter part of this paper, unless stated otherwise, we consider the above (ELL-based) definition of separability. Thus, if exists such δ_1, δ_2, $\mathbf{u_1}$, $\mathbf{u_2}$, and $\mathbf{x}^*$ that $f(\mathbf{x}^*) \leq f(\mathbf{x}^* + \delta_1\mathbf{u_1})$ and $f(\mathbf{x}^* + \delta_2\mathbf{u_2}) > f(\mathbf{x}^* + \delta_1\mathbf{u_1} + \delta_2\mathbf{u_2})$ (formula (8)), then



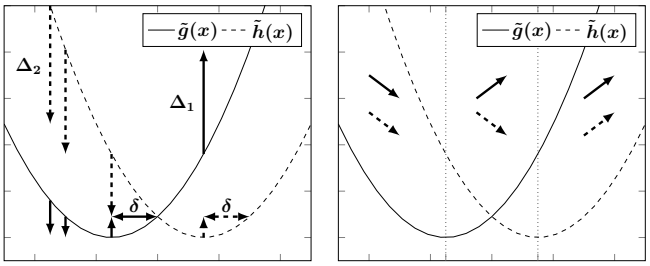
(a) Possibly separable (b) Non-separable

Fig. 2. Separability in terms of greedy optimizer behavior



(a) Differential grouping (false linkage) (b) Monotonicity checking (no false linkage)

Fig. 3. Influence of function scaling on false linkage discovery



(a) Differential grouping (high sensitivity) (b) Monotonicity checking (low sensitivity)

Fig. 4. Sensitivity to interaction discovery

X_1 and X_2 sets are interacting. Thus, monotonicity checking strategies may be classified as ELL techniques. In [20], they are also identified as strategies that do not report false linkage.

B. Decomposition Inaccuracies: False and Missing Linkage

DG-based decomposition strategies are proven that they will never report false linkage if a function is additively separable [11], [14]. A function is additively separable if every pair of its subfunctions is also additively separable. Let f_i and f_j be separable subfunctions. They are additively separable if $\forall_{x_p,x_q} \frac{\partial^2 f}{\partial x_q \partial x_p} = 0$, where x_p and x_q are arguments of subfunctions f_i and f_j, respectively [14]. Otherwise, f_i and f_j are non-additively separable[1]. If a function is non-additively separable, i.e., every pair of its subfunctions is also non-additively separable, DG-based decomposition strategies may report false linkage. Let us consider the minimization of the following function: $\bar{f}_{c,1} : [-5,5]^2 \rightarrow [0,100]$ defined as $\bar{f}_{c,1}(\mathbf{x}) = (|x_1| + |x_2|)^2$. A greedy optimizer that is deterministic (e.g., with a fixed value of the random seed) applied to optimize x_1, for any value of x_2, will converge to the same value and vice versa. If this greedy optimizer has infinite precision, it will converge to 0 for both variables. Therefore, $\bar{f}_{c,1}$ is fully separable. However, DG-based strategies will find x_1 and x_2 dependent, because using formula (2), we get $\Delta_1 = 5$ and $\Delta_2 = 7$ for $(a, a+\delta, b_1, b_2) = (1, 2, 1, 2)$.

On the other hand, monotonicity checking never reports false linkage (see Section III-A). However, in practice, due to the inaccuracy of the representation of float numbers, the appropriate value of ϵ is necessary to determine if the inequalities are satisfied (see formula (7)). Then, monotonicity checking strategies will not report false linkage also for problems with non-additively separable subproblems. DG-based strategies do not have this advantage (and also suffer from the floating-point error). However, the results presented in [11], [13], [14], [22] show that DG-based strategies are more accurate. The reason may be that monotonicity checking strategies proposed so far are more vulnerable to missing linkage. Let us consider more examples that employ function $\tilde{f}$, and its subfunctions $\tilde{g}$ and $\tilde{h}$, defined at the beginning of this subsection. We simplify formula (2) to $\tilde{g}(a+\delta) - \tilde{g}(a) \neq \tilde{h}(a+\delta) - \tilde{h}(a)$ that is equivalent to $\Delta_1 \neq \Delta_2$. Similarly, we simplify formula (3) to $\tilde{g}(a_1) \leq \tilde{g}(a_2) \wedge \tilde{h}(a_1) > \tilde{h}(a_2)$.

Fig. 3 presents another example in which DG may report false linkage. We multiply a function by a positive number, which does not change relations between its outputs, i.e., the relation between function outputs (smaller, greater, equal) for any two arguments will be the same for the original and multiplied functions. In Fig. 3a, Δ_1 and Δ_2 are indicated by vectors for various values of a spread along the horizontal axis and the same value of δ. For at least one pair (Δ_1, Δ_2) the referring vectors differ. Thus, although, the considered problem is separable, DG may detect (false) interaction between x_1 and x_2. Oppositely, monotonicity checking will consider the monotonicity intervals (Fig. 3b), which are the same for $\tilde{g}$ and $\tilde{h}$. Note that it is a necessary, but not sufficient, condition. Hence, we can use another condition, this time sufficient. If $\tilde{h}(x) = \alpha \cdot \tilde{g}(x)$, where $\alpha > 0$, then $\forall_{a_1 \neq a_1} \tilde{g}(a_1) > \tilde{g}(a_2) \vee \tilde{h}(a_1) \leq \tilde{h}(a_2)$, because $\forall_{\alpha>0} \tilde{h}(a_1) \leq \tilde{h}(a_2) \Leftrightarrow \tilde{g}(a_1) \leq \tilde{g}(a_2)$. Therefore, monotonicity checking will report x_1 and x_2 as independent.

One of the main advantages of DG is its high resistance to missing linkage. In Fig. 4a, we show that the probability of selecting a that leads to the dependency discovery is high. Monotonicity checking will always report x_1 and x_2 as dependent only if a_1 and a_2 come from the middle interval (Fig. 4b). Otherwise, the interaction may not be discovered.

Feasible values of decision variables may also influence monotonicity checking. Let us consider the decomposition of $\bar{f}_{c,2} : [-8,8] \times [-2,2] \rightarrow [0,100]$ defined as $\bar{f}_{c,2}(\mathbf{x}) = (x_1 + x_2)^2$. For $(a_1, a_2, b_1, b_2) = (-2, -1, 2, 1)$, monotonicity checking will discover the dependency between x_1 and x_2.However, if $a_1 = -6$ and $a_2 = -5$, then $\forall_{b_1,b_2 \in [-2,2]} \bar{f}_{c,2}(a_1, b_1) > \bar{f}_{c,2}(a_2, b_1) \wedge \bar{f}_{c,2}(a_1, b_2) > \bar{f}_{c,2}(a_2, b_2)$. Thus, monotonicity checking will not find the dependency for any feasible values of b_1 and b_2. Note that $\bar{f}_{c,2}$ is fully non-separable, because the value of $\bar{f}_{c,2}$ is minimal when $x_1 = -x_2$. Finally, DG-based strategies should discover

[1]See the supplementary material for examples and additional explanations.

the interaction between x_1 and x_2 easily.

IV. Incremental Recursive Ranking Grouping

Decomposition strategies based on DG and monotonicity checking may report false or missing linkage, respectively. Therefore, we propose Incremental Recursive Ranking Grouping (IRRG) employing the idea of monotonicity checking. IRRG never reports false linkage and significantly mitigates the issue of missing linkage. First, we describe Recursive Ranking Grouping (RRG) that is a key-part part of our proposition, then we present the general view of IRRG. Finally, we compare IRRG with other monotonicity checking strategies and discuss the time complexity of IRRG.

A. Basic Concept

The simplest way to decrease the level of missing linkage in monotonicity checking is to create uniformly and randomly a huge number of samples. A single sample is defined by a_1, a_2, b_1, b_2, $\mathbf{x}^*$ and by δ_1, δ_2, $\mathbf{u_1}$, $\mathbf{u_2}$, $\mathbf{x}^*$ for formulas (7) and (8), respectively. Such an approach increases the probability that at least one sample will result in dependency discovery if two variables or two sets of variables indeed interact. However, its cost may not be reasonable for the optimization process.

To increase the sensitivity of monotonicity checking, IRRG creates two rankings of samples. Each ranking consists of n_s (a user-defined parameter) samples. To check if variables x_p and x_q interact, we need b_1, b_2 ($b_1 \neq b_2$), $\mathbf{x}^*$, and n_s values that are evenly taken from the feasible set of x_p. Let us denote these n_s values as $\tilde{a}_1$, ..., $\tilde{a}_{n_s}$. Then, we create two rankings $\mathbf{r_1} = [r_{1,1}, ..., r_{1,n_s}]$ and $\mathbf{r_2} = [r_{2,1}, ..., r_{2,n_s}]$, where $r_{j,i}$ is computed for $f(\mathbf{x}^*)|_{x_p=\tilde{a}_i, x_q=b_j}$. If $\exists_{i\in\{1,...,n_s\}} r_{1,i} \neq r_{2,i}$ then the variables x_p and x_q interact. For instance, for $\bar{f}_{c,3}$: $[-3,3]^4 \rightarrow [-111,111]$ defined as $\bar{f}_{c,3}(\mathbf{x}) = (x_1 + x_2)^2 \cdot x_3 + x_4$, $p = 1$, $q = 2$, $b_1 = 1$, $b_2 = 2$, $\mathbf{x}^* = [1,0,3,2]$, and $n_s = 3$, we get $\tilde{a}_1 = -3$, $\tilde{a}_2 = 0$, and $\tilde{a}_3 = 3$. Thus, $\bar{f}_{c,3}(\mathbf{x}^*)|_{x_1=-3,x_2=1} = 14$, $\bar{f}_{c,3}(\mathbf{x}^*)|_{x_1=0,x_2=1} = 5$, $\bar{f}_{c,3}(\mathbf{x}^*)|_{x_1=3,x_2=1} = 50$, and $\bar{f}_{c,3}(\mathbf{x}^*)|_{x_1=-3,x_2=2} = 5$, $\bar{f}_{c,3}(\mathbf{x}^*)|_{x_1=0,x_2=2} = 14$, $\bar{f}_{c,3}(\mathbf{x}^*)|_{x_1=3,x_2=2} = 77$. Therefore, $\mathbf{r_1} = [2,1,3]$ and $\mathbf{r_2} = [1,2,3]$. Since $r_{1,1} \neq r_{2,1}$, x_1 and x_2 are dependent. Comparing rankings may be found equivalent to searching for a_1, a_2, b_1, b_2, and $\mathbf{x}^*$ that satisfy formula (7).

Using formula (8), we define the recursive ranking check that can detect interaction between two disjoint sets of variables X_1 and X_2. First, n_s values for each $x_j \in X_1$, namely $\tilde{a}_{1,j}$, ..., $\tilde{a}_{n_s,j}$, are evenly generated from the x_j feasible set. Then, we define n_s decision vectors $\tilde{\mathbf{x}}_\mathbf{i} = [\tilde{x}_{i,1}, ..., \tilde{x}_{i,n}]$ based on a given vector $\mathbf{x}^* = [x_1^*, ..., x_n^*]$, where

$$\tilde{x}_{i,j} = \begin{cases} \tilde{a}_{\pi_j(i),j} & , x_j \in X_1 \\ x_j^* & , x_j \notin X_1 \end{cases} \tag{9}$$

where π_j denotes a random permutation of $\{1, ..., n_s\}$ for the jth variable. These vectors are then used to generate the rankings. Using all $n_s^{|X_1|}$ available vectors is impossible due to practical reasons. The random permutation is used to overcome the possible bias. The ith value of the first ranking $r_{1,i}$ is based on $f(\tilde{\mathbf{x}}_\mathbf{i})$, whereas $r_{2,i}$ is computed for $f(\tilde{\mathbf{x}}_\mathbf{i} + \delta_2\mathbf{u_2})$. Analogously to the pair-wise interaction check, If $\exists_{i^*\in\{1,...,n_s\}} r_{1,i^*} \neq r_{2,i^*}$ then the sets X_1 and X_2 interact, because for at least one $i^* \in \{1, ..., n_s\}$, condition $f\left(\tilde{\mathbf{x}}_{\mathbf{r_1},\mathbf{i}^*}\right) \leq f\left(\tilde{\mathbf{x}}_{\mathbf{r_2},\mathbf{i}^*}\right) \wedge f\left(\tilde{\mathbf{x}}_{\mathbf{r_1},\mathbf{i}^*} + \delta_2\mathbf{u_2}\right) > f\left(\tilde{\mathbf{x}}_{\mathbf{r_2},\mathbf{i}^*} + \delta_2\mathbf{u_2}\right)$ holds. Note that this condition specifies formula (8).

B. Recursive Ranking Grouping

RRG is the main part of IRRG. It discovers dependencies between disjoint groups of variables. First, we define a set of functions that will be used in the description of RRG. A set of groups of variables that were already found interacting will be denoted as $\boldsymbol{G}$, while V is a set of variables for which no interactions were discovered yet. In matrix $\bar{\boldsymbol{X}}_\mathbf{1}$ of size $n_s \times n$, each jth column consists of n_s randomly ordered values that are evenly generated from the feasible set of the jth variable.

Function ConsiderVariables (Pseudocode 1) decides if the variables in V should be considered in the interaction check. It will happen if one of the following conditions holds. (1) $\boldsymbol{G}$ is empty, i.e., no dependencies were discovered yet (line 2). (2) V contains only one variable (line 2). (3) V is randomly divided into two disjoint subsets V_1 and V_2. The sizes of both are equal or differ by one. If we detect interaction between V_1 and V_2, then the variables in V will be considered in the interaction search (lines 6–14). (4) V interacts with at least one group from $\boldsymbol{G}$ (lines 15–21). ConsiderVariables uses CreateFirstRanking (Pseudocode 2) that creates a ranking using the concept presented in Section IV-A. Except for ranking $\mathbf{r_1}$, this function returns a vector of values $\bar{\mathbf{y}}_\mathbf{1}$ for which the ranking was computed. $\bar{\mathbf{y}}_\mathbf{1}$ is used by other functions defined in this section.

Pseudocode 1 ConsiderVariables; Checking if it is worth considering separable so far variables in the interaction search

```
   input: V: variables to check, G: groups of non-separable variables, x_hq: a high-
   quality decision vector, X̄_1: a n_s × n matrix of samples' values, x̄_2: a decision
   vector different than x_hq, f: an optimization problem, n_s: the number of samples
   output: a decision if at least one variable from V can be non-separable
1: function ConsiderVariables(V, G, x_hq, X̄_1, x̄_2, f, n_s)
2:   if |G| = 0 or |V| = 1 then
3:     return true
4:   if |V| = 0 then
5:     return false
6:   V ← shuffle V
7:   V_1 ← the first half of V
8:   V_2 ← the second half of V
9:   (ȳ_1, r_1) ← CreateFirstRanking(V_1, x_hq, X̄_1, f, n_s)
10:  if IsInteraction(V_1, V_2, x_hq, X̄_1, x̄_2, ȳ_1, r_1, f, n_s) then
11:    return true
12:  (ȳ_1, r_1) ← CreateFirstRanking(V_2, x_hq, X̄_1, f, n_s)
13:  if IsInteraction(V_2, V_1, x_hq, X̄_1, x̄_2, ȳ_1, r_1, f, n_s) then
14:    return true
15:  for i ← 1 to |G| do
16:    (ȳ_1, r_1) ← CreateFirstRanking(V, x_hq, X̄_1, f, n_s)
17:    if IsInteraction(V, G[i], x_hq, X̄_1, x̄_2, ȳ_1, r_1, f, n_s) then
18:      return true
19:    (ȳ_1, r_1) ← CreateFirstRanking(G[i], x_hq, X̄_1, f, n_s)
20:    if IsInteraction(G[i], V, x_hq, X̄_1, x̄_2, ȳ_1, r_1, f, n_s) then
21:      return true
22:  return false
```

Function IsInteraction (Pseudocode 3) is used to check the dependency between two groups of variables denoted as X_1 and X_2. To this end, we create ranking $\mathbf{r_2}$ and check if it differs from $\mathbf{r_1}$. To save computational effort, the order of ranking $\mathbf{r_2}$ is checked during its creation.

Pseudocode 2 CREATEFIRSTRANKING

input: X_1: set of variables, $\mathbf{x}$: a decision vector, $\bar{\boldsymbol{X}}_\mathbf{1}$: a $n_s \times n$ matrix of samples' values, f: an optimization problem, n_s: the number of samples
output: $\bar{\mathbf{y}}_\mathbf{1}$: values of f calculated based on $\mathbf{x}$ and $\bar{\boldsymbol{X}}_\mathbf{1}$, $\mathbf{r_1}$: the ranking of $\bar{\mathbf{y}}_\mathbf{1}$
1: **function** CREATEFIRSTRANKING($X_1, \mathbf{x}, \bar{\boldsymbol{X}}_\mathbf{1}, f, n_s$)
2: $\bar{\mathbf{x}} \leftarrow \mathbf{x}$
3: **for** $i \leftarrow 1$ **to** n_s **do**
4: $\bar{\mathbf{x}}[X_1] \leftarrow \bar{\boldsymbol{X}}_\mathbf{1}[i][X_1]$
5: $\bar{\mathbf{y}}_\mathbf{1}[i] \leftarrow f(\bar{\mathbf{x}})$ ▷ values of f calculated based on $\mathbf{x}$ and $\bar{\boldsymbol{X}}_\mathbf{1}$
6: $\mathbf{r_1} \leftarrow$ indices i from 1 to n_s ordered by $\bar{\mathbf{y}}_\mathbf{1}[i]$
7: **return** $(\bar{\mathbf{y}}_\mathbf{1}, \mathbf{r_1})$

Pseudocode 3 ISINTERACTION; Checking if two sets of variables are interacting

input: X_1, X_2: disjoint sets of variables, $\mathbf{x_{hq}}$: a high-quality decision vector, $\bar{\boldsymbol{X}}_\mathbf{1}$: a $n_s \times n$ matrix of samples' values, $\bar{\mathbf{x}}_\mathbf{2}$: a decision vector different than $\mathbf{x_{hq}}$, $\bar{\mathbf{y}}_\mathbf{1}$: values of f calculated based on $\mathbf{x_{hq}}$ and $\bar{\boldsymbol{X}}_\mathbf{1}$, $\mathbf{r_1}$: the ranking of $\bar{\mathbf{y}}_\mathbf{1}$, f: an optimization problem, n_s: the number of samples
output: a decision if X_1 and X_2 are interacting
1: **function** ISINTERACTION($X_1, X_2, \mathbf{x_{hq}}, \bar{\boldsymbol{X}}_\mathbf{1}, \bar{\mathbf{x}}_\mathbf{2}, \bar{\mathbf{y}}_\mathbf{1}, \mathbf{r_1}, f, n_s$)
2: $\bar{\mathbf{x}} \leftarrow \mathbf{x_{hq}}$
3: $\bar{\mathbf{x}}[X_2] \leftarrow \bar{\mathbf{x}}_\mathbf{2}[X_2]$
4: $\bar{\mathbf{x}}[X_1] \leftarrow \bar{\boldsymbol{X}}_\mathbf{1}[\mathbf{r_1}[1]][X_1]$
5: $\bar{\mathbf{y}}_\mathbf{2}[1] \leftarrow f(\bar{\mathbf{x}})$ ▷ values of f calculated based on $\mathbf{x_{hq}}$, $\bar{\mathbf{x}}_\mathbf{2}$, and $\bar{\boldsymbol{X}}_\mathbf{1}$
6: **for** $i \leftarrow 2$ **to** n_s **do**
7: $\epsilon_1 \leftarrow f_\gamma(\sqrt{n}+1) \cdot (|\bar{\mathbf{y}}_\mathbf{1}[\mathbf{r_1}[i]]| + |\bar{\mathbf{y}}_\mathbf{1}[\mathbf{r_1}[i-1]]|)$ ▷ formula (5)
8: **if** $\text{sgn}(\bar{\mathbf{y}}_\mathbf{1}[\mathbf{r_1}[i]] - \bar{\mathbf{y}}_\mathbf{1}[\mathbf{r_1}[i-1]], \epsilon_1) = 0$ **then** ▷ formula (10)
9: **continue**
10: $\bar{\mathbf{x}}[X_1] \leftarrow \bar{\boldsymbol{X}}_\mathbf{1}[\mathbf{r_1}[i]][X_1]$
11: $\bar{\mathbf{y}}_\mathbf{2}[i] \leftarrow f(\bar{\mathbf{x}})$
12: $\epsilon_2 \leftarrow f_\gamma(\sqrt{n}+1) \cdot (|\bar{\mathbf{y}}_\mathbf{2}[i]| + |\bar{\mathbf{y}}_\mathbf{2}[i-1]|)$ ▷ formula (5)
13: **if** $\text{sgn}(\bar{\mathbf{y}}_\mathbf{2}[i] - \bar{\mathbf{y}}_\mathbf{2}[i-1], \epsilon_2) < 0$ **then** ▷ formula (10)
14: **return** $true$
15: **return** $false$

Similarly to DG-based strategies [11]–[16], the result of this comparison of rankings may be affected by the inaccuracy of the representation of float numbers. Instead of using a user-defined ϵ, we employ the automatic estimation of its value derived from [15] (lines 7 and 12). If we denote two vectors as $\mathbf{x}_\mathbf{1}^*$ and $\mathbf{x}_\mathbf{2}^*$, then, analogously to formula (4) [15], the estimation of ϵ is computed as $\epsilon = f_\gamma(\sqrt{n}+1) \cdot (|f(\mathbf{x}_\mathbf{1}^*)| + |f(\mathbf{x}_\mathbf{2}^*)|)$ to decide which solution is better. Hence, the standard signum function may be reformulated to a version that takes ϵ into account (lines 8 and 13):

$$\text{sgn}(x, \epsilon) = \begin{cases} -1, & \text{if } x < -\epsilon \\ 0, & \text{if } |x| \leq \epsilon \\ 1, & \text{if } x > \epsilon \end{cases} \tag{10}$$

High-quality solutions occupy local optima or are close to them. Therefore, if the interaction exists, then it should reveal mainly while observing high-quality solutions [17]. To achieve this goal we start the analysis of rankings $\mathbf{r_1}$ and $\mathbf{r_2}$ from the samples of the highest quality in terms of $\mathbf{r_1}$. We believe that the best samples of different rankings may correspond to different optima if an interaction exists. Additionally, to decrease the probability that too small ϵ value causes false linkage, we discover an interaction only if both results of signum function (formula (10)) are decisive (lines 8 and 13).

To discover interactions, we use n_s values for X_1 and only two values for X_2. Thus, ISINTERACTION is not a symmetric function. We call it twice in CONSIDERVARIABLES for each pair of sets of variables we examine (lines 10, 13, 17, and 20).

In the last paragraph of Section III-B, we show that constraints, which are a part of constrained optimization problems, e.g., bounding-box constraints, may prevent monotonicity checking from discovering interactions. To this end, we prefer to use a high-quality solution $\mathbf{x_{hq}}$ that was optimized before RRG execution. The motivation behind this step is as follows. If $\mathbf{x_{hq}}$ is close to a local optimum and X_1, X_2 interact, then the following situations are possible. (1) $\mathbf{x_{hq}} + \delta_2 \mathbf{u_2}$ is not close to the local optimum, meaning that monotonicity intervals around $\mathbf{x_{hq}}$ changed and that RRG will likely detect them. (2) The monotonicity intervals around $\mathbf{x_{hq}}$ did not change, but the relations between function values around $\mathbf{x_{hq}}$ changed (see Fig. 2b). In such a situation, RRG is likely to discover interaction. (3) The monotonicity intervals and the relations of function values around $\mathbf{x_{hq}}$ remain the same. RRG will not discover interaction in this case, but such a situation is not likely if X_1 and X_2 interact. Additionally, the number of local optima also influences the RRG sensitivity. The higher number of local optima, the higher number of points from different basins of attraction that have a similar interaction discovery potential as $\mathbf{x_{hq}}$. Thus, RRG will have a higher chance of interaction discovery.

Function INTERACT (Pseudocode 4) extends function ISINTERACTION. Its objective is to search for groups of variables in $\boldsymbol{G_2}$ that interact with at least one group from $\boldsymbol{G_1}$. The output of INTERACT is a set of groups $\boldsymbol{G_1^*}$ such that $\boldsymbol{G_1} \subseteq \boldsymbol{G_1^*}$. Initially, $\boldsymbol{G_1^*}$ is a copy of $\boldsymbol{G_1}$ (line 2). To check if $\boldsymbol{G_1}$ and $\boldsymbol{G_2}$ interact, they are flatten (let $\boldsymbol{S} = \{S_1, ..., S_k\}$ be a set of k sets, then flattening $\boldsymbol{S}$ results in a set $\bigcup \boldsymbol{S} = S_1 \cup ... \cup S_k$) to $X_1 = \bigcup \boldsymbol{G_1}$ and $X_2 = \bigcup \boldsymbol{G_2}$ (lines 3 and 4), and ISINTERACTION is called (line 5). If an interaction is discovered, $\boldsymbol{G_2}$ is divided into two subsets (preferably of equal size) $\boldsymbol{G_2^1}$ and $\boldsymbol{G_2^2}$ (lines 9 and 10). INTERACT is then called recursively for $\boldsymbol{G_2^1}$ and $\boldsymbol{G_2^2}$ (lines 11 and 12). Once an interaction is found for $\boldsymbol{G_2}$ containing only one element (group), then this single group is added to $\boldsymbol{G_1^*}$ (line 7).

Pseudocode 4 INTERACT; Searching for groups of variables that are interacting, but were not discovered yet

input: $\boldsymbol{G_1}$, $\boldsymbol{G_2}$: disjoint sets of groups of non-separable variables, $\mathbf{x_{hq}}$: a high-quality decision vector, $\bar{\boldsymbol{X}}_\mathbf{1}$: a $n_s \times n$ matrix of samples' values, $\bar{\mathbf{x}}_\mathbf{2}$: a decision vector different than $\mathbf{x_{hq}}$, $\bar{\mathbf{y}}_\mathbf{1}$: values of f calculated based on $\mathbf{x_{hq}}$ and $\bar{\boldsymbol{X}}_\mathbf{1}$, $\mathbf{r_1}$: the ranking of $\bar{\mathbf{y}}_\mathbf{1}$, f: an optimization problem, n_s: the number of samples
output: $\boldsymbol{G_1^*}$: the union of $\boldsymbol{G_1}$ and some groups from $\boldsymbol{G_2}$ that interact with at least one group from $\boldsymbol{G_1}$
1: **function** INTERACT($\boldsymbol{G_1}, \boldsymbol{G_2}, \mathbf{x_{hq}}, \bar{\boldsymbol{X}}_\mathbf{1}, \bar{\mathbf{x}}_\mathbf{2}, \bar{\mathbf{y}}_\mathbf{1}, \mathbf{r_1}, f, n_s$)
2: $\boldsymbol{G_1^*} \leftarrow \boldsymbol{G_1}$
3: $X_1 \leftarrow$ flatten $\boldsymbol{G_1}$
4: $X_2 \leftarrow$ flatten $\boldsymbol{G_2}$
5: **if** ISINTERACTION($X_1, X_2, \mathbf{x_{hq}}, \bar{\boldsymbol{X}}_\mathbf{1}, \bar{\mathbf{x}}_\mathbf{2}, \bar{\mathbf{y}}_\mathbf{1}, \mathbf{r_1}, f, n_s$) **then**
6: **if** $|\boldsymbol{G_2}| = 1$ **then**
7: Add the group from $\boldsymbol{G_2}$ to $\boldsymbol{G_1^*}$
8: **else**
9: $\boldsymbol{G_2^1} \leftarrow$ the first half of $\boldsymbol{G_2}$
10: $\boldsymbol{G_2^2} \leftarrow$ the second half of $\boldsymbol{G_2}$
11: $\boldsymbol{G_1^{1*}} \leftarrow$ INTERACT($\boldsymbol{G_1}, \boldsymbol{G_1^1}, \mathbf{x_{hq}}, \bar{\boldsymbol{X}}_\mathbf{1}, \bar{\mathbf{x}}_\mathbf{2}, \bar{\mathbf{y}}_\mathbf{1}, \mathbf{r_1}, f, n_s$)
12: $\boldsymbol{G_1^{2*}} \leftarrow$ INTERACT($\boldsymbol{G_1}, \boldsymbol{G_2^2}, \mathbf{x_{hq}}, \bar{\boldsymbol{X}}_\mathbf{1}, \bar{\mathbf{x}}_\mathbf{2}, \bar{\mathbf{y}}_\mathbf{1}, \mathbf{r_1}, f, n_s$)
13: Add all new groups from $\boldsymbol{G_1^{1*}}$ to $\boldsymbol{G_1^*}$
14: Add all new groups from $\boldsymbol{G_1^{2*}}$ to $\boldsymbol{G_1^*}$
15: **return** $\boldsymbol{G_1^*}$

The general procedure of RRG is presented in Pseudocode 5. RRG requires providing the interaction matrix Θ, which single element $\theta_{p,q} \in \{0, 1\}$ informs if two variables, x_p and x_q, are dependent. Θ may be represented by a graph.

Finding the connected components of this graph leads to identifying groups of interacting variables [13]. Considering remarks to ISINTERACTION presented above, we assume that at least one provided decision vector is of high quality. First, using Θ, we create groups of interacting variables. The order of variables within groups is random to remove possible bias (lines 5–10). Then, according to the description presented in Section IV-A, we create $\bar{\boldsymbol{X}}_{\mathbf{1}}$ (lines 11–14). Using CONSIDERVARIABLES we decide if variables that were not found interacting yet, shall take part in the interaction discovery (line 16). We shuffle $\boldsymbol{G}$ to remove possible bias and create $\boldsymbol{G_1}$ and $\boldsymbol{G_2}$ (lines 17–19).

Pseudocode 5 RRG; Recursive Ranking Grouping

```
   input: x_hq: a high-quality decision vector, x̄_2: a decision vector different than
   x_hq, Θ: the interaction matrix, f: an optimization problem, n: the problem size,
   lb: the lower bounds of f, ub: the upper bounds of f, n_s: the number of samples
   output: NonSeps: groups of non-separable variables
1: function RRG(x_hq, x̄_2, Θ, f, n, lb, ub, n_s)
2:   NonSeps ← ∅
3:   G ← ∅                          ▷ groups of non-separable variables to detect new linkage
4:   V ← variables from x_1 to x_n
5:   for v ∈ V do
6:     Inters ← extract variables interacting with v from Θ
7:     if |Inters| > 1 then
8:       V ← V − Inters                                          ▷ the difference of sets
9:       Inters ← shuffle Inters
10:      Add Inters to G as a group
11:  for i ← 1 to n do                  ▷ building a n × n_s matrix of samples' values
12:    X̄_1^T[i] ← n_s values evenly taken from [lb[i], ub[i]]
13:    X̄_1^T[i] ← shuffle X̄_1^T[i]
14:  X̄_1 ← transpose X̄_1^T                          ▷ a n_s × n matrix of samples' values
15:  if CONSIDERVARIABLES(V, G, x_hq, X̄_1, x̄_2, f, n_s) then
16:    Add all variables from V to G as single groups
17:  G ← shuffle the order of groups in G
18:  G_1 ← take the first group from G
19:  G_2 ← the other groups from G
20:  while |G_2| > 0 do
21:    X_1 ← flatten G_1
22:    (ȳ_1, r_1) ← CREATEFIRSTRANKING(X_1, x_hq, X̄_1, f, n_s)
23:    G_1^* ← INTERACT(G_1, G_2, x_hq, X̄_1, x̄_2, ȳ_1, r_1, f, n_s)
24:    if |G_1| = |G_1^*| then
25:      if |G_1| = 1 then
26:        minSize = min_i |G_2[i]|
27:        if |G_1[1]| ≥ max{minSize, 2} then
28:          Remove half variables from G_1[1]
29:        else
30:          G_1 ← take the first group from G_2
31:          G_2 ← G_2 excluding the first group
32:      else
33:        Add flattened G_1 to NonSeps
34:        G_1 ← take the first group from G_2
35:        G_2 ← G_2 excluding the first group
36:    else
37:      G_1 ← G_1^*
38:      G_2 ← G_2 excluding groups from G_1
39:  if |G_1| > 1 then
40:    Add flattened G_1 to NonSeps
41:  return NonSeps
```

The latter part of RRG focuses on interaction discovery (lines 20–40). We create ranking $\mathbf{r_1}$ and call INTERACT to find new interactions between groups of variables from $\boldsymbol{G_1}$ and $\boldsymbol{G_2}$. If interactions were found, we add dependent groups to $\boldsymbol{G_1}$, remove them from $\boldsymbol{G_2}$ (lines 37 and 38) and call INTERACT again for the extended $\boldsymbol{G_1}$. If no new interactions were found, but $\boldsymbol{G_1}$ was extended and contains more than one group, then $\boldsymbol{G_1}$ is flattened and added as a single group to the output set of groups of interacting variables, $\boldsymbol{G_1}$ is replaced with a single group that is excluded from $\boldsymbol{G_2}$ (it is equivalent to the random choice, because $\boldsymbol{G_2}$ was shuffled) (lines 33–35). If no extensions to the group initially inserted to $\boldsymbol{G_1}$ were discovered (i.e. $\boldsymbol{G_1}$ contains only one group after calling INTERACT), then we randomly remove half of the variables from the single group in $\boldsymbol{G_1}$ (lines 26–28). The intuition behind this step is as follows. If $|\bigcup \boldsymbol{G_1}|$ is large, then some subgroup of variables in $\bigcup \boldsymbol{G_1}$ may influence rankings $\mathbf{r_1}$ and $\mathbf{r_2}$ so significantly that the influence of the variables in $\bigcup \boldsymbol{G_2}$ will become negligible. Thus, if we remove some of the variables from a single group in $\boldsymbol{G_1}$ randomly, then some of the overwhelmed variables in $\bigcup \boldsymbol{G_2}$ may gain their influence on $\mathbf{r_1}$ and $\mathbf{r_2}$ rankings, resulting in the interaction discovery and mitigating missing linkage. Usually, the situation described above occurs when most interactions are discovered. Thus, it seems reasonable to decrease the size of a single group in $\boldsymbol{G_1}$ only if its size is not less than the size of the shortest group in $\boldsymbol{G_2}$. The above procedure mitigates missing linkage and does not seem computationally expensive.

C. Incremental Grouping

IRRG builds the interaction matrix Θ incrementally. The general IRRG procedure is presented in Pseudocode 6. First, a user-defined optimizer is executed to find $\mathbf{x_{hq}}$ (a high-quality solution) (line 2), and the interaction matrix Θ is initialized as reporting no interactions (line 3). Then, RRG is called to get $\boldsymbol{NonSepsTmp}$—the set of groups of interacting variables (line 9). This set is used to update Θ by inserting the newly discovered interactions (line 10). The Θ updates ensure transitivity. These operations are repeated until RRG will not find any new interaction in ϵ_{sti} (a user-defined parameter) iterations in a row or if RRG did not discover any new interactions during the first iteration (line 13). The latter part of this condition was introduced to decrease the computational cost of IRRG. The motivation is as follows. The lower is the number of interactions in Θ, the easier is to supplement it. Thus, if RRG can not discover any new interactions in the first iteration, then repeating it does not seem reasonable.

IRRG returns two sets of variable groups. $\boldsymbol{NonSeps}$ contains groups of interacting variables, whereas in $\boldsymbol{Seps}$, separable variables are joined into groups of a user-defined size ϵ_s (lines 26–32). This step is adopted from RDG3 [16] to improve separable variables processing by CC frameworks.

IRRG can be tuned by adjusting values of four parameters: an initial optimizer, the number of samples n_s, the stale iteration threshold ϵ_{sti}, and the preferred size of groups of separable variables ϵ_s. The optimizer is executed to find a high-quality solution $\mathbf{x_{hq}}$. In case of LSGO problems, especially partially separable ones, we recommend exploring the search space first and then exploiting the found promising region. Otherwise, high-quality solutions for some subproblems may not be found. However, for some problems, the initial optimization will not be helpful and will not lead to obtaining better decomposition. The cost of this initial optimization should be reasonable, e.g., at most $2 \cdot n_s \cdot n$ FFEs, which corresponds to the lowest time complexity of IRRG. The higher the values of n_s and ϵ_{sti}, the higher the

Pseudocode 6 Incremental Recursive Ranking Grouping

input: f: an optimization problem, n: the problem size, $\mathbf{lb}$: the lower bounds of f, $\mathbf{ub}$: the upper bounds of f, $optimizer$: using during the initial optimization, n_s: the number of samples, ϵ_s: the preferred size of groups of separable variables, ϵ_{sti}: the stale iterations threshold
output: $\boldsymbol{Seps}$: groups of separable variables, $\boldsymbol{NonSeps}$: groups of non-separable variables

1: $\boldsymbol{Seps}, \boldsymbol{NonSeps} \leftarrow \varnothing$
2: $\mathbf{x_{hq}} \leftarrow$ optimize f using $optimizer$
3: $\Theta \leftarrow n \times n$ identity matrix ▷ the interaction matrix
4: $firstIter \leftarrow true$ ▷ marks the first iteration of IRRG
5: $cnt_{sti} \leftarrow 0$ ▷ the counter of the consecutive stale iterations
6: $terminate \leftarrow false$
7: **while not** $terminate$ **do**
8: $\mathbf{x_{lq}} \leftarrow$ create a random solution
9: $\boldsymbol{NonSepsTmp} \leftarrow$ RRG($\mathbf{x_{hq}}, \mathbf{x_{lq}}, \Theta, f, n, \mathbf{lb}, \mathbf{ub}, n_s$)
10: Update Θ taking groups from $\boldsymbol{NonSepsTmp}$ into account
11: **if** any new interactions were not discovered **then** ▷ see line 10
12: $cnt_{sti} \leftarrow cnt_{sti} + 1$
13: $terminate \leftarrow firstIter$ **or** $cnt_{sti} = \epsilon_{sti}$
14: **else**
15: $cnt_{sti} \leftarrow 0$
16: $firstIter \leftarrow false$
17: $V \leftarrow$ variables from x_1 to x_n
18: $S \leftarrow \varnothing$ ▷ separable variables before their grouping into $\boldsymbol{Seps}$
19: **for** $v \in V$ **do**
20: $Inters \leftarrow$ extract variables interacting with v from Θ
21: **if** $|Inters| = 1$ **then**
22: Add v to S
23: **else**
24: Add $Inters$ to $\boldsymbol{NonSeps}$ as a group
25: $V \leftarrow V - Inters$ ▷ the difference of sets
26: **while** $|S| > 0$ **do**
27: **if** $|S| < \epsilon_s$ **then**
28: Add S to $\boldsymbol{Seps}$ as a group
29: $S \leftarrow \varnothing$
30: **else**
31: Add ϵ_s variables from S to $\boldsymbol{Seps}$ as a group
32: Delete ϵ_s variables from S
33: **return** $(\boldsymbol{Seps}, \boldsymbol{NonSeps})$

probability of obtaining better decomposition. However, the decomposition cost increases at the same time. The discussion about ϵ_s can be found in [16].

D. Comparison with Other Monotonicity Checking Strategies

IRRG and FVIL are recursive monotonicity checking strategies. Both use formula (7) to decide if two groups of variables (X_1 and X_2) are interacting. If the interaction is discovered, X_2 is divided into two (nearly) equally sized groups, X_2^1 and X_2^2. Then, X_1 is checked if it interacts with X_2^1 or X_2^2. FVIL creates a new sample (δ_1, δ_2, $\mathbf{u_1}$, $\mathbf{u_2}$, $\mathbf{x}^*$) for each iteration check randomly. Consequently, if X_1 interacts with X_2 because variables $x_p \in X_1$ and $x_q \in X_2$ are interacting and x_q is a member of X_2^1 afterwards, then the interaction between X_1 and X_2^1 could not be discovered because of different samples. Therefore, in IRRG, values assigned to variables in X_1, X_2^1, and X_2^2 are the same as in the previous step. Due to repeating value assignments, some FFEs may be cached and reused in the next recursive steps.

Another difference that causes FVIL to be more vulnerable to missing linkage is the size of X_1. In FVIL, X_1 is always a single-element group. In IRRG, variables found dependent are added to X_1 in subsequent interaction checks. Subsequent iterations of IRRG also try to extend the already found groups of interacting variables. Such interaction checks using larger X_1 decrease the probability of missing linkage. Finally, FVIL does not apply the initial optimization. As explained in Section IV-B, using $\mathbf{x_{hq}}$ may significantly mitigate missing linkage for some optimization problems.

Some monotonicity checking strategies employ an optimizer during interaction search [19], [20], e.g., Cooperative Coevolution with Variable Interaction Learning (CCVIL) [20]. CCVIL executes an optimizer for each variable in each interaction discovery phase. However, the optimizer is executed only for one iteration, and its population is limited to only three individuals. Thus, we may expect that $\mathbf{x_{hq}}$ employed by IRRG is of significantly higher quality.

E. Time Complexity

IRRG is a recursive decomposition strategy, similarly to RDG-based strategies [14]–[16]. The typical time complexity (considering FFEs) for this class is $\mathcal{O}(n \log(n))$. In this paper, we consider LSGO, thus, we may expect that for large n, the cost of initial optimization (held to obtain $\mathbf{x_{hq}}$) is negligible. Thus, we ignore this part of IRRG in the analysis presented below. We refer to the analogous analysis of RDG [14]. The creation of $\mathbf{r_1}$ ranking costs n_s FFEs in each IRRG iteration. The cost of a single INTERACT execution is not higher than n_s FFEs (the analogous operation in RDG requires three FFEs [14]). Thus, the cost of decomposing n-dimensional problem in the first iteration of IRRG is as follows.

1) Fully separable problems—about $n_s \cdot n + n_s \cdot n$ FFEs.
2) Fully non-separable problems—at most $2 \cdot n_s \cdot n + n_s$ FFEs.
3) Partially separable problems with m subproblems of size l—at most $2 \cdot n_s \cdot n \cdot \log_2(n) + n_s \cdot m$ FFEs.
4) Partially separable problems with one non-separable subproblem of size l—at most $n_s \cdot (n - l) + 2 \cdot n_s \cdot l \cdot \log_2(n) + n_s \cdot (n - l + 1)$ FFEs.
5) Rosenbrock function [23] that is an overlapping problem—about $2 \cdot n_s \cdot n \cdot \log_2(n) + n_s \cdot n/2$ FFEs.

For consecutive IRRG iterations, we propose the similar analysis. Let us denote the number of variables that were not found interacting yet as n_V and the number of already discovered groups of interacting variables as $n_{\boldsymbol{G}}$. Thus, $n_V + n_{\boldsymbol{G}} < n$. If all possible interactions were already discovered, then similar to fully separable problems, approximately $2 \cdot n_s \cdot (n_V + n_{\boldsymbol{G}})$ additional FFEs are needed. Otherwise, since $n_V + n_{\boldsymbol{G}} < n$, each consecutive IRRG iteration will still use less FFEs than the first iteration.

RRG includes the operation of removing half of variables from the single group belonging to $\boldsymbol{G_1}$ (Pseudocode 5, line 28). The number of variable removals will be the highest when $|\bigcup \boldsymbol{G_1}| = n - 1 \wedge |\bigcup \boldsymbol{G_2}| = 1$ and will not exceed $\log_2(n-1)$. The cost of each cut is not higher than $2 \cdot n_s$ FFEs, n_s FFEs for the creation of ranking $\mathbf{r_1}$ and another n_s FFEs for INTERACT. Thus, the total cost of each cut is below $2 \cdot n_s \cdot \log_2(n-1)$ FFEs. In the worst scenario, the cost of CONSIDERVARIABLES resulting from calling ISINTERACTION and creating ranking $\mathbf{r_1}$ twice for single variables and twice for each group of interacting variables is approximately $4 \cdot n_s + 4 \cdot n_s \cdot n_{\boldsymbol{G}}$ FFEs. Therefore, if $n_s \ll n$ and the

number of all IRRG iterations is much smaller than n, the time complexity of IRRG is $\mathcal{O}(n\log(n))$.

One of the cases with a high number of IRRG iterations is as follows. The considered problem consists of $n/2$ subproblems of size 2, one interaction is found during the first iteration, and at every ϵ_{sti}th iteration another single interation is found. Then, the number of IRRG iterations is $1+\epsilon_{sti}\cdot n/2$. Since the time complexity of finding a new single interaction is $\mathcal{O}(n)$, the worst time complexity of IRRG is $\mathcal{O}(n^2)$.

V. Experiments and Results Analysis

The objective of the experiments was twofold. First—to check if the proposed problem decomposition strategy is more robust than the DG-based strategies. The second objective was to verify if IRRG can be successfully embedded into CC frameworks. To this end, we integrate IRRG with CCBC [8], [10] and CCFR2 [9], which were shown effective in solving LSGO problems. We compare these methods denoted as CBCC-IRRG and CCFR2-IRRG with their versions employing RDG3 denoted as CBCC-RDG3 [8], [10], [16] and CCFR2-RDG3 [9], [16], respectively. The competing methods were supplemented by SHADE-ILS [6]. Note that CBCC-RDG3 and SHADE-ILS are the state-of-the-art methods when LSGO is considered. To the best of our knowledge, CCFR2-RDG3 was not proposed yet. Nevertheless, considering it among the competing methods seems justified and desirable. IRRG- and RDG3-embedded CCs significantly outperformed CBCC-FVIL [8], [22] and CCFR2-FVIL [9], [22]. Therefore, FVIL-based results are reported in the supplementary material.

This section is divided into four subsections. First, we present the setup of all considered optimization methods. Then, we describe the chosen test problems. In Section V-C, we discuss the decomposition accuracy of IRRG and RDG3 as well as their decomposition costs. Finally, the optimization results are reported in the last subsection.

A. Optimization Methods' Setup

CC frameworks require component optimizers. For this purpose, we employ CMA-ES [41], which was shown to be the best option for both considered CC frameworks [9], [14]. We use CMA-ES implementation[2], which was shown the most efficient in [42]. The original implementation of CBCC-RDG3 was supported in Matlab[3]. It was rewritten to Python to integrate it with the source code of CMA-ES. The implementation of CCFR2 was made based on its pseudocodes presented in [9]. For RDG3, we use its original Matlab source code. The decomposition that resulted from using this source code was used as an input to the Python implementations of CBCC and CCFR2. Similarly, IRRG was written in C++, but its output was passed to Python. The pack with the source code, the detailed results of all runs, and the results of all performed statistical tests can be downloaded from the public repository[4]. For SHADE-ILS, we took the source code provided by its Authors[5].

[2]https://github.com/CMA-ES/pycma
[3]https://bitbucket.org/yuans/rdg3
[4]https://github.com/kommar/IRRG
[5]https://github.com/dmolina/shadeils

The parameters of the considered decomposition strategies, CC frameworks, and SHADE-ILS are adopted from papers in which they were proposed or recently applied. For RDG3, we use $\epsilon_s = 100$ and $\epsilon_n = 50$ [16]. CBCC is executing CMA-ES for 1000 FFEs and the smoothing factor w is 0.5 [10], [16]. For CCFR2, we use $w = 0.1$ and CMA-ES (executed for 100 iterations) as a component optimizer [9]. SHADE-ILS uses the population size of 100 for its underlying differential evolution, MTS-LS1 [35] with the initial step set to 20, the minimum improvement ratio is 5% and the allowed number of iterations without the improvement is 3 [6].

The configuration of the proposed IRRG is as follows. To obtain $\mathbf{x_{hq}}$, we use two optimizers. First, we use SHADE with the recommended settings [34] (population size: 100; arc ratio: 2; $p$best ratio: 0.1; memory size: 10^3). Then, the solution proposed by SHADE is optimized by MTS-LS1 [35] (search range: 20% of the difference between the upper bound and the lower bound). The motivation behind using two optimizers is to consider SHADE as a global search optimizer and MTS-LS1 as a local search optimizer. They are executed for $5\cdot 10^3$ and $1.5\cdot 10^4$ FFEs, respectively. The other IRRG parameters are: $\epsilon_{sti} = 15$, $n_s = 10$, and $\epsilon_s = 100$ (the value of ϵ_s is adopted from [16]). The FFE-based stop conditions and parameters ϵ_{sti} and n_s were tuned to find their minimal values that still lead to the perfect decomposition of considered non-overlapping benchmarks.

B. Test Problems

As benchmarks we use functions from IEEE CEC'2013 special session on LSGO [23]. CEC'2013 was also considered by other up-to-date papers that considered LSGO [6], [9], [13]–[16]. However, in CEC'2013 almost all separable subfunctions are additively separable, which favors DG-based strategies (see Section III-B). To this end, we propose two new sets based on CEC'2013, which eliminate this drawback.

Let us consider a function f and its range Y. Let g be a function that is strictly increasing over Y. Then, we can state that $\forall_{\mathbf{x_1^*},\mathbf{x_2^*}\in\Omega} f(\mathbf{x_1^*}) < f(\mathbf{x_2^*}) \Rightarrow h(\mathbf{x_1^*}) < h(\mathbf{x_2^*})$, where $h = g\circ f$. For each such function composition, any result of a comparison between solutions does not change. Therefore, f and h have the same intervals of monotonicity, optima, and variable interactions (see Section III-A). However, even if f is additively separable, h may be non-additively separable, e.g., when g is a non-linear function. Nevertheless, their ideal interaction matrices are the same.

The minimal value of all fifteen CEC'2013 functions is not lower than 0. Thus, as g we need functions that are strictly increasing over non-negative arguments. In this paper, we use $g_1(y) = y^2$ and $g_2(y) = \sqrt{y}$. They were chosen, because they increase in different ways. The test cases created using g_1 and g_2 are denoted as $(f_i)^2$ and $\sqrt{f_i}$, respectively. For the sake of clarity, we define a set of the ith functions F_i as $\{f_i, (f_i)^2, \sqrt{f_i}\}$ for $i \in \{1, ..., 15\}$. To summarize, we consider the following five categories of functions [9], [23].

1) Fully separable functions (F_1–F_3).
2) Partially separable functions with separable variables (F_4–F_7).

3) Partially separable functions with no separable variables (F_8–F_{11}).
4) Overlapping functions (F_{12}–F_{14}).
5) Fully non-separable functions (F_{15}).

Considering the type of interaction between separable subfunctions, we split the first four categories into:

1) Functions with additively separable subfunctions (f_1, f_2, and f_4–f_{14}).
2) Functions with non-additively separable subfunctions (f_3, $(f_1)^2$–$(f_{14})^2$, and $\sqrt{f_1}$–$\sqrt{f_{14}}$).

The computation budget was set to $3 \cdot 10^6$ FFEs and each experiment was repeated 25 times [6], [9], [13]–[16], [23].

As a real-world optimization problem with non-additively separable subproblems, we analyze a multi-path routing problem in computer and communication networks. In contrast to the traditional routing approach that transmits all traffic of a given demand along a single path, multi-path routing splits the traffic among several paths. Multi-path routing can improve network efficiency in terms of various measures of Quality of Service, including survivability, congestion and throughput [43]–[45]. More details, including the results, can be found in the supplementary material.

C. Accuracy and Costs of Decomposition

IRRG is a decomposition strategy that never reports false linkage (in contrast to DG-based strategies) and should not suffer from missing linkage, which is a typical drawback of monotonicity checking. We employ three measures proposed in [46] to verify the accuracy of the decomposition. The first measure (ρ_1) returns the accuracy of finding interaction, and its value is maximum when there is no missing linkage. The second measure (ρ_2) refers to false linkage. The higher its value is, the lower is the number of non-existing interactions detected. The value of the third measure (ρ_3) is maximal when there is neither false nor missing linkage.

$$\rho_1 = \frac{\sum_{i=1}^{n}\sum_{j=1,j\neq i}^{n}(\Theta \circ \Theta^*)_{i,j}}{\sum_{i=1}^{n}\sum_{j=1,j\neq i}^{n}(\Theta^*)_{i,j}} \cdot 100\% \tag{11}$$

$$\rho_2 = \frac{\sum_{i=1}^{n}\sum_{j=1,j\neq i}^{n}((\mathbf{1}-\Theta) \circ (\mathbf{1}-\Theta^*))_{i,j}}{\sum_{i=1}^{n}\sum_{j=1,j\neq i}^{n}(\mathbf{1}-\Theta^*)_{i,j}} \cdot 100\% \tag{12}$$

$$\rho_3 = \frac{\sum_{i=1}^{n}\sum_{j=1,j\neq i}^{n}(\mathbf{1}-|\Theta-\Theta^*|)_{i,j}}{n\cdot(n-1)/2} \cdot 100\% \tag{13}$$

where n is the problem size, Θ^* is the ideal interaction matrix, $\mathbf{1}$ denotes the all-ones matrix, $\circ$ indicates the Hadamard product of two matrices, and $|\Theta - \Theta^*|$ is a matrix which elements are defined as $|(\Theta)_{i,j} - (\Theta^*)_{i,j}|$.

In Table I, we present the decomposition accuracy of IRRG and RDG3. IRRG is non-deterministic, thus, we report the summarized results of 50 runs. We summarize the runs of CBCC-IRRG and CCFR2-IRRG to show the quality of decomposition that was later on used during the optimization. For all non-overlapping functions, IRRG is highly repeatable—the standard deviation is equal to 0% for these test cases. For functions with separable variables, ρ_2 and ρ_3 were computed just before joining them into groups of ϵ_s size (otherwise, IRRG and RDG3 would report false linkage). For the overlapping functions the ρ_2 values are below 100% for IRRG, although it never reports false linkage. This is caused by the fact that ρ_1, ρ_2, and ρ_3 allow only for direct interactions, while IRRG and RDG3 consider also the indirect ones [12], [47].

For the separable and partially separable functions (additive or non-additive), the values of all measures are optimal for all IRRG runs, i.e., IRRG found all interactions and never reported false linkage regardless of the type of interaction between subfunctions (Table I, F_1–F_{11}). For the standard CEC'2013 functions, except for f_3 and f_6, RDG3 did not report false linkage. Such results were expected for f_3, because it is the only function from the standard CEC'2013 set that is non-additively separable. Thus, $\rho_2 = \rho_3 = 0\%$ for RDG3 reported for f_3 mean that all separable variables were grouped together. On the other hand, f_6 is additively separable, which makes $\rho_2 = 71.93\%$ an unexpected result. The possible reason behind this result seems to be the inaccuracy of automatic ϵ estimation procedure, i.e., in some cases the value of ϵ might be too low. Additionally, for f_5, f_8, and f_{10} the value of ρ_1 is below 100%, which means that RDG3 did not discover all interactions. The most convincing explanation seems to be too high value of ϵ. To omit the issue of missing linkage, IRRG executes RRG several times, which increases the risk of reporting false linkage (in the case when ϵ is too low). Nevertheless, Table I shows that considering only those samples for which the outcome of f differs, in terms of the ϵ value, is sufficient to report false linkage never or very rarely.

All modified f_1–f_3 functions are non-additively separable. Thus, we expect that for these functions ρ_2 is 0% for RDG3. The results for $(f_1)^2$ and $\sqrt{f_1}$ have significantly higher values (although still much worse than the optimal 100%). These results are caused by cases when too high value of ϵ allows RDG3 to omit discovering some of the false dependencies. Moreover, RDG3 tries to limit the size of a single group to ϵ_n, which may mitigate false linkage phenomenon. Note that the inappropriate value of ϵ influences RDG3 differently in $(f_1)^2$ and $\sqrt{f_1}$ cases. For the modified versions of partially separable functions, we also expect grouping all variables together by RDG3, i.e., $\rho_1 = 100\%$ and $\rho_2 = 0\%$. However, similarly to $(f_1)^2$–$(f_3)^2$ and $\sqrt{f_1}$–$\sqrt{f_3}$, too high value of ϵ may lead to mitigating the false linkage phenomenon.

A perfect, unique decomposition does not exist for overlapping problems (F_{12}–F_{14}), and it may be beneficial to use various decompositions during the optimization process [17]. In all overlapping test cases, every variable is dependent (directly or indirectly) on all the others. For these problems, the value of ρ_1 is maximum when only one group is created, while ρ_2 is maximum if all created groups consist of only directly dependent variables. Finally, ρ_3 is a trade-off between ρ_1 and ρ_2. Thus, its value cannot reach 100%. Therefore, each IRRG execution may lead to various groupings and the standard deviation values for F_{12}–F_{14} are higher than 0%.

IRRG can detect both types of separability (additive and non-additive). Thus, we expect it to return similar results for each overlapping CEC'2013 function and its modifications. The results reported in Table I confirm this expectation—for most of the functions the values of all three measures are sim-

TABLE I
DECOMPOSITION ACCURACY

Func	IRRG									RDG3		
	ρ_1			ρ_2			ρ_3			ρ_1	ρ_2	ρ_3
	Med	Avg	Std	Med	Avg	Std	Med	Avg	Std			
f_1	N/A	N/A	N/A	**100%**	**100%**	**0%**	**100%**	**100%**	**0%**	N/A	**100%**	**100%**
$(f_1)^2$	N/A	N/A	N/A	**100%**	**100%**	**0%**	**100%**	**100%**	**0%**	N/A	39.97%	39.97%
$\sqrt{f_1}$	N/A	N/A	N/A	**100%**	**100%**	**0%**	**100%**	**100%**	**0%**	N/A	50.40%	50.40%
f_2	N/A	N/A	N/A	**100%**	**100%**	**0%**	**100%**	**100%**	**0%**	N/A	**100%**	**100%**
$(f_2)^2$	N/A	N/A	N/A	**100%**	**100%**	**0%**	**100%**	**100%**	**0%**	N/A	0%	0%
$\sqrt{f_2}$	N/A	N/A	N/A	**100%**	**100%**	**0%**	**100%**	**100%**	**0%**	N/A	0%	0%
f_3	N/A	N/A	N/A	**100%**	**100%**	**0%**	**100%**	**100%**	**0%**	N/A	0%	0%
$(f_3)^2$	N/A	N/A	N/A	**100%**	**100%**	**0%**	**100%**	**100%**	**0%**	N/A	0%	0%
$\sqrt{f_3}$	N/A	N/A	N/A	**100%**	**100%**	**0%**	**100%**	**100%**	**0%**	N/A	0%	0%
f_4	**100%**	**100%**	**0%**	**100%**	**100%**	**0%**	**100%**	**100%**	**0%**	**100%**	**100%**	**100%**
$(f_4)^2$	**100%**	**100%**	**0%**	**100%**	**100%**	**0%**	**100%**	**100%**	**0%**	89.26%	56.57%	57.13%
$\sqrt{f_4}$	**100%**	**100%**	**0%**	**100%**	**100%**	**0%**	**100%**	**100%**	**0%**	70.70%	73.09%	73.05%
f_5	**100%**	**100%**	**0%**	**100%**	**100%**	**0%**	**100%**	**100%**	**0%**	98.85%	**100%**	99.98%
$(f_5)^2$	**100%**	**100%**	**0%**	**100%**	**100%**	**0%**	**100%**	**100%**	**0%**	89.22%	51.78%	52.42%
$\sqrt{f_5}$	**100%**	**100%**	**0%**	**100%**	**100%**	**0%**	**100%**	**100%**	**0%**	89.83%	83.17%	83.29%
f_6	**100%**	**100%**	**0%**	**100%**	**100%**	**0%**	**100%**	**100%**	**0%**	**100%**	71.93%	72.42%
$(f_6)^2$	**100%**	**100%**	**0%**	**100%**	**100%**	**0%**	**100%**	**100%**	**0%**	70.13%	67.36%	67.41%
$\sqrt{f_6}$	**100%**	**100%**	**0%**	**100%**	**100%**	**0%**	**100%**	**100%**	**0%**	84.86%	58.02%	58.48%
f_7	**100%**	**100%**	**0%**	**100%**	**100%**	**0%**	**100%**	**100%**	**0%**	**100%**	**100%**	**100%**
$(f_7)^2$	**100%**	**100%**	**0%**	**100%**	**100%**	**0%**	**100%**	**100%**	**0%**	**100%**	92.62%	92.74%
$\sqrt{f_7}$	**100%**	**100%**	**0%**	**100%**	**100%**	**0%**	**100%**	**100%**	**0%**	**100%**	92.62%	92.74%
f_8	**100%**	**100%**	**0%**	**100%**	**100%**	**0%**	**100%**	**100%**	**0%**	70.20%	**100%**	97.98%
$(f_8)^2$	**100%**	**100%**	**0%**	**100%**	**100%**	**0%**	**100%**	**100%**	**0%**	63.88%	99.72%	97.28%
f_8	**100%**	**100%**	**0%**	**100%**	**100%**	**0%**	**100%**	**100%**	**0%**	69.57%	38.80%	40.89%
f_9	**100%**	**100%**	**0%**	**100%**	**100%**	**0%**	**100%**	**100%**	**0%**	**100%**	**100%**	**100%**
$(f_9)^2$	**100%**	**100%**	**0%**	**100%**	**100%**	**0%**	**100%**	**100%**	**0%**	84.28%	44.36%	47.07%
$\sqrt{f_9}$	**100%**	**100%**	**0%**	**100%**	**100%**	**0%**	**100%**	**100%**	**0%**	92.47%	63.37%	65.35%
f_{10}	**100%**	**100%**	**0%**	**100%**	**100%**	**0%**	**100%**	**100%**	**0%**	93.28%	**100%**	99.54%
$(f_{10})^2$	**100%**	**100%**	**0%**	**100%**	**100%**	**0%**	**100%**	**100%**	**0%**	93.21%	98.82%	98.44%
$\sqrt{f_{10}}$	**100%**	**100%**	**0%**	**100%**	**100%**	**0%**	**100%**	**100%**	**0%**	89.85%	98.84%	98.23%
f_{11}	**100%**	**100%**	**0%**	**100%**	**100%**	**0%**	**100%**	**100%**	**0%**	**100%**	**100%**	**100%**
$(f_{11})^2$	**100%**	**100%**	**0%**	**100%**	**100%**	**0%**	**100%**	**100%**	**0%**	98.60%	43.58%	47.32%
$\sqrt{f_{11}}$	**100%**	**100%**	**0%**	**100%**	**100%**	**0%**	**100%**	**100%**	**0%**	95.09%	49.35%	52.45%
f_{12}	**99.20%**	**99.07%**	**0.32%**	84.86%	83.94%	4.80%	84.89%	83.97%	4.79%	98.10%	**95.28%**	**95.29%**
$(f_{12})^2$	99.20%	99.14%	0.29%	**84.56%**	**81.80%**	**9.14%**	**84.59%**	**81.83%**	**9.12%**	**100%**	0%	0.20%
$\sqrt{f_{12}}$	99.20%	99.14%	0.36%	**83.54%**	**81.77%**	**7.51%**	**83.57%**	**81.80%**	**7.49%**	**100%**	0%	0.20%
f_{13}	**99.93%**	**99.84%**	**0.22%**	0.23%	18.04%	23.88%	8.44%	24.78%	21.90%	92.65%	**98.61%**	**98.12%**
$(f_{13})^2$	99.93%	99.83%	0.18%	**0.23%**	**21.33%**	**24.36%**	**8.44%**	**27.79%**	**22.34%**	**100%**	0%	8.23%
$\sqrt{f_{13}}$	99.85%	99.81%	0.20%	**0.23%**	**21.77%**	**26.30%**	**8.44%**	**28.20%**	**24.12%**	**100%**	0%	8.23%
f_{14}	**99.70%**	**99.70%**	**0.24%**	37.54%	33.07%	28.28%	42.66%	38.55%	25.93%	94.55%	**96.88%**	**96.68%**
$(f_{14})^2$	**99.78%**	**99.78%**	**0.22%**	13.35%	22.66%	25.41%	20.47%	29.01%	23.30%	87.83%	**53.26%**	**56.11%**
$\sqrt{f_{14}}$	**99.89%**	**99.78%**	**0.24%**	0.35%	24.52%	27.28%	8.55%	30.72%	25.01%	73.85%	**56.69%**	**58.11%**
f_{15}	**100%**	**100%**	**0%**	N/A	N/A	N/A	**100%**	**100%**	**0%**	**100%**	N/A	**100%**
$(f_{15})^2$	**100%**	**100%**	**0%**	N/A	N/A	N/A	**100%**	**100%**	**0%**	**100%**	N/A	**100%**
$\sqrt{f_{15}}$	**100%**	**100%**	**0%**	N/A	N/A	N/A	**100%**	**100%**	**0%**	**100%**	N/A	**100%**

ilar for the standard, squared, and square rooted function versions. This observation was verified by the unpaired Wilcoxon test (significance level of 5%) with the Holm-Bonferroni correction for measures ρ_1–ρ_3 and pairs $\left(f_i, (f_i)^2\right)$, $\left(f_i, \sqrt{f_i}\right)$, and $\left((f_i)^2, \sqrt{f_i}\right)$, where $i \in \{12, 13, 14\}$. Such results of statistical tests may be surprising for f_{14} for which the differences between the median values of ρ_1 seem high. Nevertheless, it is justified by the high value of the standard deviation.

For RDG3, the observations for F_{12}–F_{14} are opposite to the observations for IRRG. For f_{12}–f_{14}, RDG3 is able to find a balance between missing and false linkage, i.e. the values of all measures are above 90%. However, for the modified versions of f_{12} and f_{13}, the values of ρ_2 are equal to 0%. For $(f_{14})^2$ and $\sqrt{f_{14}}$, the results differ most probably due to the influence of the inappropriate value of ϵ for some interaction checks and the impact of ϵ_n.

Finally, both decomposition strategies handle the fully non-separable function f_{15}. In this case, the proposed modifications does not affect the quality of the decomposition.

The results presented above show that IRRG outperforms RDG3 in decomposing the fully and partially separable functions. For fully non-separable functions, both strategies perform equally well, whereas for the overlapping functions the

TABLE II
FFEs CONSUMED BY DECOMPOSITION STRATEGIES

Func	RDG3		IRRG (Avg) / RDG3 (Avg)				
	Avg	Std	Min	Max	Med	Avg	Std
f_1–f_3	4.0E+03	1.7E+03	6.7	13.3	13.3	11.1	3.8
f_4–f_7	1.0E+04	1.1E+03	4.8	8.8	7.8	7.3	1.8
f_8–f_{11}	1.9E+04	2.5E+02	**3.6**	5.4	4.2	4.3	0.8
f_{12}–f_{14}	2.7E+04	1.9E+04	4.2	**14.6**	4.4	7.7	5.9
f_{15}	6.0E+03	N/A	4.7	4.7	4.7	4.7	N/A
$(f_1)^2$–$(f_3)^2$	6.3E+03	5.3E+02	5.8	6.7	6.7	6.4	0.5
$(f_4)^2$–$(f_7)^2$	8.1E+03	1.7E+03	5.6	12.7	10.0	9.6	3.1
$(f_8)^2$–$(f_{11})^2$	1.4E+04	5.4E+03	**3.8**	9.3	6.8	6.7	2.6
$(f_{12})^2$–$(f_{14})^2$	6.5E+03	1.5E+03	8.3	**124.2**	12.7	48.4	65.7
$(f_{15})^2$	6.0E+03	N/A	4.8	4.8	4.8	4.8	N/A
$\sqrt{f_1}$–$\sqrt{f_3}$	6.4E+03	7.9E+02	5.4	6.7	6.7	6.3	0.7
$\sqrt{f_4}$–$\sqrt{f_7}$	1.0E+04	3.1E+03	5.4	12.8	7.2	8.1	3.5
$\sqrt{f_8}$–$\sqrt{f_{11}}$	1.2E+04	5.0E+03	**3.8**	13.5	8.2	8.4	4.1
$\sqrt{f_{12}}$–$\sqrt{f_{14}}$	6.3E+03	1.1E+03	8.9	**123.0**	12.8	48.2	64.8
$\sqrt{f_{15}}$	6.0E+03	N/A	4.9	4.9	4.9	4.9	N/A

TABLE III
IRRG'S NUMBER OF WINS, TIES, AND LOSSES AGAINST RDG3 ON THE BASIS OF CBCC AND CCFR2

i	CBCC-IRRG vs. CBCC-RDG3			CCFR2-IRRG vs. CCFR2-RDG3		
	f_i w/t/l	$(f_i)^2$ w/t/l	$\sqrt{f_i}$ w/t/l	f_i w/t/l	$(f_i)^2$ w/t/l	$\sqrt{f_i}$ w/t/l
1–3	1/**2**/0	**3**/0/0	**3**/0/0	1/**2**/0	**3**/0/0	**3**/0/0
4–7	0/**4**/0	**3**/0/1	**3**/1/0	0/**4**/0	**3**/0/1	**3**/1/0
8–11	0/**2**/**2**	**4**/0/0	**4**/0/0	1/**3**/0	**3**/1/0	**3**/1/0
12–14	1/0/**2**	**2**/1/0	1/**2**/0	1/0/**2**	**2**/1/0	**3**/0/0
15	0/**1**/0	0/**1**/0	0/**1**/0	0/**1**/0	0/**1**/0	0/**1**/0
Total	2/**9**/4	**12**/2/1	**11**/4/0	3/**10**/2	**11**/3/1	**12**/3/0

TABLE IV
CCFR2-IRRG'S NUMBER OF WINS, TIES, AND LOSSES AGAINST CBCC-IRRG AND SHADE-ILS

i	CCFR2-IRRG vs. CBCC-IRRG			CCFR2-IRRG vs. SHADE-ILS		
	f_i w/t/l	$(f_i)^2$ w/t/l	$\sqrt{f_i}$ w/t/l	f_i w/t/l	$(f_i)^2$ w/t/l	$\sqrt{f_i}$ w/t/l
1–3	0/**3**/0	0/**3**/0	0/**3**/0	0/0/**3**	0/0/**3**	0/1/**2**
4–7	0/**4**/0	0/**4**/0	0/**4**/0	**3**/0/1	**3**/0/1	**3**/0/1
8–11	**2**/**2**/0	1/**3**/0	**2**/**2**/0	**3**/1/0	**3**/1/0	**2**/**2**/0
12–14	0/**3**/0	0/**3**/0	0/**3**/0	0/1/**2**	0/0/**3**	0/1/**2**
15	0/**1**/0	0/**1**/0	0/**1**/0	0/0/**1**	0/0/**1**	0/0/**1**
Total	2/**13**/0	1/**14**/0	2/**13**/0	6/2/**7**	6/1/**8**	5/4/**6**

situation is indecisive. Hence, we can state that, in general, IRRG proposes the higher quality decomposition than RDG3. However, this advantage has its costs—as shown in Table II for the standard CEC'2013 functions, IRRG consumes from 3.6 up to even 14.6 times more FFEs on average. For the modified functions the overhead is similar except the overlapping functions, which are up to 124.2 times more expensive than for RDG3. The results of the Student's t-test with the Holm-Bonferroni correction confirm that the differences between the RDG3 and IRRG costs are statistically significant.

For F_8 and F_{10} functions, the cost of IRRG is significantly different for their standard and modified versions. Since IRRG is insensitive to the modifications, it seems these differences were caused by the ϵ value estimation procedure.

IRRG consumed the most FFEs for F_{12}, i.e., 7.4E+05 on average. Functions in F_{12} are overlapping, and their subfunction sizes are equal to 2. Thus, if two subfunctions overlap, there is one shared variable. IRRG tends to join all overlapping subproblems into one group. Therefore, to join two overlapping subfunctions, there is only one possible interaction between two variables that must be discovered. Since functions in F_{12} consist of many overlapping subfunctions and discovering dependency between two variables is hard for IRRG in this case, many iterations of IRRG are necessary.

The maximum number of IRRG iterations was reached for $\sqrt{f_{12}}$ and was equal to 169 whereas only one iteration was needed for F_1-F_3. On average IRRG executed RRG 22.74 ± 29.47 times. After excluding the fully separable functions and F_{12}, which are special cases for IRRG, the average number of iterations is 19.22 ± 4.44. These average values are much smaller than considered problem sizes. Thus, the typical time complexity of IRRG may be found as $\mathcal{O}(n \log(n))$.

D. Optimization Results

Decomposition of a problem is only a part of the whole optimization process. Additionally, even the high-quality decomposition does not guarantee the successful optimization of a given problem. Optimization methods that can use such decomposition wisely are also needed [17]. For instance, CBCC and CCFR2 offer a speed-up of a CC framework by greedily choosing the most promising subproblem to optimize [8]–[10]. Another example is joining separable variables into groups and optimizing them by CMA-ES [16], which can effectively solve fully separable problems up to 100 variables [48]. Thus, in this section, we compare the effectiveness of CBCC and CCFR2 after embedding IRRG and RDG3. Additionally, we also consider SHADE-ILS, which is dedicated to solving LSGO problems, but does not use problem decomposition. The observed differences in the results quality of the considered optimization methods are confirmed by the unpaired Wilcoxon test with the Holm-Bonferroni correction method at the significance level of 5%.

Table III presents the comparison of the effectiveness of CBCC and CCFR2 after embedding IRRG and RDG3. For the standard CEC'2013 functions, the effectiveness of both considered frameworks is similar for both embedded decomposition strategies. The situation changes significantly when the modified CEC'2013 functions are considered. For both sets of modified CEC'2013 functions, CBCC-IRRG and CCFR2-IRRG outperform the RDG3-embedded CC frameworks significantly. Thus, we may state that embedding IRRG into the state-of-the-art CC frameworks is a more robust proposition than RDG3. Although IRRG consumes more FFEs than RDG3, its cost is still only a small part of the overall computational budget. Therefore, the higher cost of IRRG does not influence the quality of results for f_1–f_{15}.

In Table IV, we compare the effectiveness of CCFR2-IRRG to CBCC-IRRG and SHADE-ILS. The results show that the effectiveness of CCFR2-IRRG and CBCC-IRRG is highly similar. CCFR2-IRRG was chosen for the comparison to

SHADE-ILS, because it performs slightly better. SHADE-ILS is significantly different from CC-based optimizers, and it does not rely on problem decomposition. As expected, for partially separable problems from the standard and modified CEC'2013 sets, CCFR2-IRRG outperforms SHADE-ILS significantly. However, the situation is the opposite for fully separable, overlapping, and non-separable functions. Such results are a consequence of the construction of CBCC and CCFR2. Additionally, SHADE-ILS employs MTS-L1, a local search optimizer dedicated to fully separable problems [6], [35]. MTS-L1 along with the restart mechanism seems to be the reason why SHADE-ILS has outperformed all CC-based optimizers for F_1–F_3. Note that CMA-ES (employed by all of the considered CC-based optimizers) is dedicated to optimizing problems with at most 100 variables [48], while all of the considered fully non-separable functions are high-dimensional. Finally, all functions in the CEC'2013 set are homogeneous, i.e., all their subfunctions derive from the same base function (e.g., Rastrigin's function) [23]. Such a feature may favor SHADE-ILS that adjusts the parameter values of its underlying SHADE and the choice of a local search method (MTS-LS1 or L-BFGS-B) to fitness landscape characteristics. If considered problems are not homogeneous, then such problems should be more suitable for CC-based optimizers because they may adjust their behaviour to the subset of interacting variables not to the whole variable set.

Usually, when IRRG-embedded CC frameworks outperform SHADE-ILS, the order of magnitude of optimization result differences is higher when compared to the opposite situations. In supplementary material, we support an extended discussion on this issue and Fig. S-1 to picture it.

In Table V, we present the results for the standard CEC'2013 functions. These detailed results reveal some phenomena discussed below. According to Table I, only IRRG decomposes f_8 precisely. Therefore, CCFR2-IRRG produced about 10^8 and 10^{16} times better results than RDG3-embedded CCs and SHADE-ILS, respectively. However, CBCC-IRRG proposed results of quality that was similar to RDG3-embedded CCs. The reason seems to be as follows. CBCC and CCFR2 focus on optimizing those components that have the highest impact on the global fitness. However, in CCFR2, the component's initial relatively poor contribution becomes insignificant faster. Thus, CCFR2 may spend more time improving those initially less significant components. This example shows that even having the perfect decomposition of a partially separable problem may not be enough to reach the high-quality results. The differences between CBCC and CCFR2 are also visible for f_{11}, for which both CCFR2 versions outperform the CBCC versions.

For overlapping problems, the perfect decomposition may not exist [13], [17]. Therefore, the structure of the proposed groups influenced the quality of the final results the most significantly. f_{13} and f_{14} are defined as overlapping problems with conforming and conflicting subproblems, respectively [49]. When subproblems are conforming, shared variables (variables that belong to more than one subproblem) may be optimized for each subproblem separately, i.e., the optimal value of a shared variable is the same for all subproblems the shared variable belongs to. If subproblems are conflicting, optimizing one subproblem collides with the optimization of other subproblems that share the same variables. Note that the considered CC-based optimizers process disjoint components and each shared variable must belong to only one component. Therefore, if subproblems are conforming, it could be better to use smaller components like optimizing each subproblem separately. If subproblems contradict, then it may be better to use larger components to optimize many conflicting subproblems at the same time. In general, RDG3 tends to create more groups containing fewer variables than IRRG due to the ϵ_n threshold. Therefore, RDG3 seems suitable for f_{13} (RGD3-embedded CCs report approximately 100 times better

TABLE V
RESULTS FOR THE STANDARD CEC'2013 SET

Func	Stats	CBCC-IRRG	CCFR2-IRRG	CBCC-RDG3	CCFR2-RDG3	SHADE-ILS
f_1	Med	7.87E-19	6.78E-19	8.14E-19	7.64E-19	**0.00E+00**
	Avg	9.16E-19	7.41E-19	8.57E-19	8.89E-19	**2.40E-28**
	Std	3.35E-19	1.89E-19	2.77E-19	3.07E-19	**5.10E-28**
f_2	Med	2.35E+03	2.35E+03	2.36E+03	2.33E+03	**1.03E+03**
	Avg	2.34E+03	2.36E+03	2.34E+03	2.33E+03	**1.06E+03**
	Std	9.64E+01	1.24E+02	9.56E+01	9.57E+01	**1.37E+02**
f_3	Med	2.02E+01	2.02E+01	2.04E+01	2.04E+01	**2.01E+01**
	Avg	2.02E+01	2.02E+01	2.04E+01	2.04E+01	**2.01E+01**
	Std	1.07E-01	1.07E-01	7.48E-02	6.89E-02	**1.13E-02**
f_4	Med	**1.58E+04**	**2.08E+03**	**5.68E+03**	**8.74E+02**	2.52E+08
	Avg	**1.97E+04**	**1.62E+04**	**1.45E+04**	**1.77E+04**	2.54E+08
	Std	**2.00E+04**	**3.03E+04**	**2.54E+04**	**3.46E+04**	9.98E+07
f_5	Med	2.42E+06	2.22E+06	2.23E+06	2.09E+06	**1.31E+06**
	Avg	2.32E+06	2.17E+06	2.31E+06	2.06E+06	**1.29E+06**
	Std	4.97E+05	3.23E+05	3.64E+05	3.28E+05	**2.04E+05**
f_6	Med	**9.96E+05**	9.96E+05	**9.96E+05**	**9.96E+05**	1.04E+06
	Avg	**1.00E+06**	1.01E+06	**9.99E+05**	**1.00E+06**	1.04E+06
	Std	**2.27E+04**	2.27E+04	**1.34E+04**	**2.25E+04**	5.64E+03
f_7	Med	**9.43E-22**	**9.25E-22**	**9.18E-22**	**9.19E-22**	1.07E+02
	Avg	**9.33E-22**	**9.27E-22**	**9.38E-22**	**9.25E-22**	1.18E+03
	Std	**9.62E-23**	**7.24E-23**	**8.06E-23**	**6.11E-23**	4.94E+03
f_8	Med	9.13E+03	**3.50E-05**	5.91E+03	3.48E+03	8.65E+11
	Avg	9.12E+03	**4.47E-05**	7.58E+03	3.41E+03	9.32E+11
	Std	1.37E+03	**2.99E-05**	5.23E+03	1.40E+03	5.80E+11
f_9	Med	**1.59E+08**	**1.50E+08**	**1.51E+08**	**1.65E+08**	**1.68E+08**
	Avg	**1.64E+08**	**1.49E+08**	**1.56E+08**	**1.67E+08**	**1.64E+08**
	Std	**2.76E+07**	**3.19E+07**	**3.05E+07**	**3.23E+07**	**2.31E+07**
f_{10}	Med	**9.05E+07**	**9.05E+07**	**9.05E+07**	**9.05E+07**	9.28E+07
	Avg	**9.10E+07**	**9.17E+07**	**9.12E+07**	**9.23E+07**	9.27E+07
	Std	**1.27E+06**	**1.88E+06**	**1.54E+06**	**2.08E+06**	4.27E+05
f_{11}	Med	5.49E-12	**6.70E-18**	9.64E-15	**7.11E-18**	4.99E+05
	Avg	1.01E-10	**1.73E-17**	2.47E-12	**7.13E-18**	5.11E+05
	Std	1.56E-10	**3.36E-17**	9.11E-12	**3.47E-18**	1.37E+05
f_{12}	Med	8.12E+02	9.04E+02	7.24E+02	7.88E+02	**6.46E-01**
	Avg	2.32E+03	9.06E+02	7.10E+02	7.98E+02	**6.60E+01**
	Std	7.17E+03	7.18E+01	9.01E+01	7.76E+01	**2.38E+02**
f_{13}	Med	1.34E+06	1.50E+06	4.65E+04	**1.75E+04**	1.04E+06
	Avg	1.14E+06	1.30E+06	5.02E+04	**2.43E+04**	1.13E+06
	Std	5.61E+05	5.80E+05	2.40E+04	**1.66E+04**	1.00E+06
f_{14}	Med	1.60E+07	1.53E+07	1.47E+09	2.15E+09	**7.55E+06**
	Avg	1.89E+07	2.01E+07	1.82E+09	2.38E+09	**7.69E+06**
	Std	8.75E+06	1.14E+07	1.52E+09	1.70E+09	**1.13E+06**
f_{15}	Med	2.22E+06	2.27E+06	2.20E+06	2.18E+06	**6.10E+05**
	Avg	2.21E+06	2.29E+06	2.19E+06	2.27E+06	**7.88E+05**
	Std	2.35E+05	1.91E+05	1.72E+05	2.52E+05	**7.62E+05**

results than IRRG-embedded ones). The situation is opposite for f_{14}. Finally, for f_{12}, which contains neither conforming nor contradicting subproblems, both variable grouping ways seem equally useful. Thus, the differences in the optimization results for f_{13} and f_{14} arise from the differences of the underlying problem structure and the employed grouping manner.

In Tables S-I and S-II (in the supplementary material, Section S-II), we report the results for the modified CEC'2013 sets. As expected, for these functions, RDG3-embedded CCs are significantly less effective when compared to IRRG-embedded ones. The results remain similar for $(f_{15})^2$ and $\sqrt{f_{15}}$, which is expected because these functions are fully non-separable. RDG3-embedded CCs are no longer dominating for $(f_{13})^2$ and $\sqrt{f_{13}}$, because RDG3 detects more (false) dependencies. Therefore, it proposes larger groups, which may not be advantageous for overlapping problems with conforming subproblems. However, despite increasing the size of the groups, RDG3-embedded CCs do not improve their effectiveness for $(f_{14})^2$ and $\sqrt{f_{14}}$ containing conflicting subfunctions. Indeed, RDG3 proposes larger groups for these functions, but since they contain false linkage, these groups mix variables from different subfunctions. Such low-quality groups do not improve the quality of the search.

Functions from F_5, i.e., f_5, $(f_5)^2$, and $\sqrt{f_5}$, contain separable variables. However, differently to other functions of this type, the quality of the results reported for F_5 is similar for all IRRG- and RDG3-embedded CCs. Based on the reported results, it seems that for F_5, the influence on the fitness of separable variables is significantly higher than for the other partially separable functions with separable subproblems (F_4, F_6, F_7). Therefore, the quality of the decomposition does not play an important role in the optimization of f_5, $(f_5)^2$, and $\sqrt{f_5}$. Note that from all the considered optimization methods, SHADE-ILS is the most effective in optimizing fully separable problems. Hence, SHADE-ILS outperformed the others for F_5.

VI. Conclusion

In this article, we propose a new decomposition strategy, namely IRRG, which derives from monotonicity checking, never reports false linkage, and significantly mitigates the risk of missing linkage. Although IRRG is more expensive than RDG3, its typical time complexity is also $\mathcal{O}(n\log(n))$. Additionally, the FFE cost of IRRG is low when compared to the overall optimization cost. IRRG- and RDG3-embedded CCs report similar quality results for the standard CEC'2013 functions, which are largely suitable for RDG3. However, for both proposed modified CEC'2013 sets with functions with non-additively separable subfunctions, the effectiveness of IRRG-embedded CCs remains the same, while for RDG3-embedded CCs deteriorates significantly. Moreover, IRRG-embedded CCs outperform RDG3-embedded CCs for the considered real-world optimization problem with non-additively separable subproblems[6].

The comparison of IRRG-embedded CCs and SHADE-ILS shows that CCs may choose different component optimizers depending on the component size and the existence of separable variables. Another promising future work direction is adjusting the ϵ_{sti} value during the IRRG run and further improvements in the procedure of automatic ϵ estimation. Finally, proposing new benchmark sets with non-homogeneous functions may be helpful for future research.

[6]See the supplementary material.

Acknowledgment

The authors wish to express their gratitude to Roza Goscien for valuable discussions on real-world optimization problems.

References

[1] J.-R. Jian, Z.-H. Zhan, and J. Zhang, "Large-scale evolutionary optimization: a survey and experimental comparative study," *Int. J. Mach. Learn. Cybern.*, vol. 11, 2020.

[2] Y.-H. Jia, Y. Mei, and M. Zhang, "A memetic level-based learning swarm optimizer for large-scale water distribution network optimization," in *Proc. Genet. Evol. Comput. Conf.*, 2020, p. 1107–1115.

[3] A. Yaman, D. C. Mocanu, G. Iacca, G. Fletcher, and M. Pechenizkiy, "Limited evaluation cooperative co-evolutionary differential evolution for large-scale neuroevolution," in *Proc. Genet. Evol. Comput. Conf.*, 2018, p. 569–576.

[4] Z.-H. Zhan, L. Shi, K. Tan, and J. Zhang, "A survey on evolutionary computation for complex continuous optimization," *Artif. Intell. Rev.*, 2021.

[5] W. Dong, T. Chen, P. Tiňo, and X. Yao, "Scaling up estimation of distribution algorithms for continuous optimization," *IEEE Trans. Evol. Comput*, vol. 17, no. 6, pp. 797–822, 2013.

[6] D. Molina, A. LaTorre, and F. Herrera, "Shade with iterative local search for large-scale global optimization," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, 2018, pp. 1–8.

[7] M. A. Potter and K. A. De Jong, "A cooperative coevolutionary approach to function optimization," in *Parallel Problem Solving Nat.*, 1994, pp. 249–257.

[8] M. N. Omidvar, X. Li, and X. Yao, "Smart use of computational resources based on contribution for cooperative co-evolutionary algorithms," in *Proc. Genet. Evol. Comput. Conf.*, 2011, p. 1115–1122.

[9] M. Yang, A. Zhou, C. Li, J. Guan, and X. Yan, "Ccfr2: A more efficient cooperative co-evolutionary framework for large-scale global optimization," *Inf. Sci.*, vol. 512, pp. 64 – 79, 2020.

[10] Y. Sun, M. Kirley, and X. Li, "Cooperative co-evolution with online optimizer selection for large-scale optimization," in *Proc. Genet. Evol. Comput. Conf.*, 2018, p. 1079–1086.

[11] M. N. Omidvar, X. Li, Y. Mei, and X. Yao, "Cooperative co-evolution with differential grouping for large scale optimization," *IEEE Trans. Evol. Comput*, vol. 18, no. 3, pp. 378–393, 2014.

[12] Y. Sun, M. Kirley, and S. K. Halgamuge, "Extended differential grouping for large scale global optimization with direct and indirect variable interactions," in *Proc. Genet. Evol. Comput. Conf.*, 2015, p. 313–320.

[13] M. N. Omidvar, M. Yang, Y. Mei, X. Li, and X. Yao, "Dg2: A faster and more accurate differential grouping for large-scale black-box optimization," *IEEE Trans. Evol. Comput*, vol. 21, no. 6, pp. 929–942, 2017.

[14] Y. Sun, M. Kirley, and S. K. Halgamuge, "A recursive decomposition method for large scale continuous optimization," *IEEE Trans. Evol. Comput*, vol. 22, no. 5, pp. 647–661, 2018.

[15] Y. Sun, M. N. Omidvar, M. Kirley, and X. Li, "Adaptive threshold parameter estimation with recursive differential grouping for problem decomposition," in *Proc. Genet. Evol. Comput. Conf.*, 2018, p. 889–896.

[16] Y. Sun, X. Li, A. Ernst, and M. N. Omidvar, "Decomposition for large-scale optimization problems with overlapping components," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, 2019, pp. 326–333.

[17] M. W. Przewozniczek and M. M. Komarnicki, "Empirical linkage learning," *IEEE Trans. Evol. Comput*, vol. 24, no. 6, pp. 1097–1111, 2020.

[18] M. W. Przewozniczek, B. Frej, and M. M. Komarnicki, "On measuring and improving the quality of linkage learning in modern evolutionary algorithms applied to solve partially additively separable problems," in *Proc. Genet. Evol. Comput. Conf.*, 2020, p. 742–750.

[19] K. Weicker and N. Weicker, "On the improvement of coevolutionary optimizers by learning variable interdependencies," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, vol. 3, 1999, pp. 1627–1632.

[20] W. Chen, T. Weise, Z. Yang, and K. Tang, "Large-scale global optimization using cooperative coevolution with variable interaction learning," in *Parallel Problem Solving Nat.*, 2010, pp. 300–309.
[21] L. Sun, S. Yoshida, X. Cheng, and Y. Liang, "A cooperative particle swarm optimizer with statistical variable interdependence learning," *Inf. Sci.*, vol. 186, no. 1, pp. 20–39, 2012.
[22] H. Ge, L. Sun, X. Yang, S. Yoshida, and Y. Liang, "Cooperative differential evolution with fast variable interdependence learning and cross-cluster mutation," *Appl. Soft Comput.*, vol. 36, pp. 300–314, 2015.
[23] X. Li, K. Tang, M. N. Omidvar, Z. Yang, and K. Qin, "Benchmark functions for the cec'2013 special session and competition on large-scale global optimization," 2013.
[24] M. N. Omidvar, X. Li, and K. Tang, "Designing benchmark problems for large-scale continuous optimization," *Inf. Sci.*, vol. 316, pp. 419 – 436, 2015.
[25] F. van den Bergh and A. Engelbrecht, "A cooperative approach to particle swarm optimization," *IEEE Trans. Evol. Comput*, vol. 8, no. 3, pp. 225–239, 2004.
[26] Z. Yang, K. Tang, and X. Yao, "Large scale evolutionary optimization using cooperative coevolution," *Inf. Sci.*, vol. 178, no. 15, pp. 2985–2999, 2008.
[27] M. N. Omidvar, X. Li, and X. Yao, "Cooperative co-evolution with delta grouping for large scale non-separable function optimization," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, 2010, pp. 1–8.
[28] S. Mahdavi, M. E. Shiri, and S. Rahnamayan, "Cooperative co-evolution with a new decomposition method for large-scale optimization," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, 2014, pp. 1285–1292.
[29] Y. Wang, H. Liu, F. Wei, T. Zong, and X. Li, "Cooperative coevolution with formula-based variable grouping for large-scale global optimization," *Evol. Comput.*, vol. 26, no. 4, pp. 569–596, 2018.
[30] P. L. Toint, "Test problems for partially separable optimization and results for the routine pspmin," *The University of Namur, Department of Mathematics, Belgium, Tech. Rep*, 1983.
[31] "Ieee standard for floating-point arithmetic," *IEEE Std 754-2008*, pp. 1–70, 2008.
[32] A. LaTorre, S. Muelas, and J.-M. Peña, "A comprehensive comparison of large scale global optimizers," *Inf. Sci.*, vol. 316, pp. 517–549, 2015.
[33] A. Hadi, A. Wagdy, and K. Jambi, "Lshade-spa memetic framework for solving large-scale optimization problems," *Complex Intell. Syst.*, vol. 5, 2018.
[34] R. Tanabe and A. Fukunaga, "Success-history based parameter adaptation for differential evolution," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, 2013, pp. 71–78.
[35] Lin-Yu Tseng and Chun Chen, "Multiple trajectory search for large scale global optimization," in *Proc. IEEE Congr. Evol. Comput. CEC (IEEE World Congr. Comput. Intell.)*, 2008, pp. 3052–3059.
[36] J. L. Morales and J. Nocedal, "Remark on "algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound constrained optimization"," vol. 38, no. 1, 2011.
[37] O. Mersmann, M. Preuss, and H. Trautmann, "Benchmarking evolutionary algorithms: Towards exploratory landscape analysis," in *Parallel Problem Solving Nat.*, 2010, pp. 73–82.
[38] A. Engelbrecht, P. Bosman, and K. Malan, "The influence of fitness landscape characteristics on particle swarm optimisers," *Nat. Comput.*, 2021.
[39] P. Kerschke and H. Trautmann, "Automated Algorithm Selection on Continuous Black-Box Problems by Combining Exploratory Landscape Analysis and Machine Learning," *Evol. Comput.*, vol. 27, no. 1, pp. 99–127, 2019.
[40] S. Chen, A. Bolufé-Röhler, J. Montgomery, and T. Hendtlass, "An analysis on the effect of selection on exploration in particle swarm optimization and differential evolution," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, 2019, pp. 3037–3044.
[41] N. Hansen and A. Ostermeier, "Completely derandomized self-adaptation in evolution strategies," *Evol. Comput.*, vol. 9, no. 2, pp. 159–195, 2001.
[42] R. Biedrzycki, "On equivalence of algorithm's implementations: The cma-es algorithm and its five implementations," in *Proc. Genet. Evol. Comput. Conf. Companion*, 2019, p. 247–248.
[43] M. Pioro and D. Medhi, *Routing, Flow, and Capacity Design in Communication and Computer Networks*. Morgan Kaufmann, 2004.
[44] I. Cidon, R. Rom, and Y. Shavitt, "Analysis of multi-path routing," *IEEE/ACM Trans. Networking*, vol. 7, no. 6, pp. 885–896, 1999.
[45] R. Banner and A. Orda, "Multipath routing algorithms for congestion minimization," *IEEE/ACM Trans. Networking*, vol. 15, no. 2, pp. 413–424, 2007.
[46] Y. Mei, M. N. Omidvar, X. Li, and X. Yao, "A competitive divide-and-conquer algorithm for unconstrained large-scale black-box optimization," *ACM Trans. Math. Softw.*, vol. 42, no. 2, 2016.
[47] M. W. Przewozniczek, M. M. Komarnicki, and B. Frej, "Direct linkage discovery with empirical linkage learning," in *Proc. Genet. Evol. Comput. Conf.*, 2021, p. 609–617.
[48] R. Ros and N. Hansen, "A simple modification in cma-es achieving linear time and space complexity," in *Parallel Problem Solving Nat.*, 2008, pp. 296–305.
[49] Y.-H. Jia, Y. Mei, and M. Zhang, "Contribution-based cooperative co-evolution for nonseparable large-scale problems with overlapping subcomponents," *IEEE Trans. Cybern.*, pp. 1–14, 2020.

Marcin Michal Komarnicki received the B.S. and M.S. degrees in computer science from the Wroclaw University of Science and Technology, Wrocław, Poland, in 2013 and 2014, respectively, where he is currently pursuing the Ph.D. degree.

His current research interests include large scale global optimization, linkage learning techniques, and multi-objective optimization.

Michal Witold Przewozniczek received the Ph.D., and D.Sc. degrees in computer science from the Wroclaw University of Science and Technology, Wrocław, Poland, in 2012 and 2021, respectively.

He is an Associate Professor with the Wroclaw University of Science and Technology. His research interests concentrate on evolutionary optimization development and its application in practice. This includes problem decomposition via linkage learning techniques, using problem structure decomposition in Gray- and Black-box optimization, multi-population approaches, and dynamic management of subpopulations. He is also a co-owner of the company implementing recent scientific achievements in the industry field. More details at www.przewozniczek.eu.

Halina Kwasnicka received her PhD and D.Sc. (Habilitation) degrees from Wroclaw University of Science and Technology (WUST) in 1980 and 2000, respectively. In 2009 she was nominated by the President of Poland as a full professor (the highest scientific title in Poland).

Currently, she is a full professor at the Department of Artificial Intelligence at Wroclaw University of Science and Technology, Poland. Her research interest focuses on a broad understanding of Artificial Intelligence, from nature-inspired methods, data mining, and Explainable Artificial Intelligence. She co-authored over 200 scientific papers, five books (in Polish) and was co-editor of two books (Springer). She gave several invited lectures. She was awarded as TOP of the TOP Women in AI 2022 by Perspektywy Women in Tech and received the "Distinguished Mentor in AI" award.

Krzysztof Walkowiak received his Ph.D. and D.Sc. (Habilitation) degrees from Wrocław University of Science and Technology (WroclawTech) in 2000 and 2008, respectively. He is currently a full professor in the Department of Systems and Computer Networks, WroclawTech.

His research interest focuses on optimization of communication networks, machine learning, metaheuristics, and intent-based networks. He received the 2014 Fabio Neri Best Paper Award and Best Paper Awards of DRCN 2009 and RNDM 2015. He has co-authored above 300 papers, 70 of them published in JCR journals.