國立臺灣大學電機資訊學院電機工程學系

碩士論文

Department of Electrical Engineering
College of Electrical Engineering and Computer Science
National Taiwan University

Master's Thesis

實數變數離散化研究：應用於實數最佳化之離散模型建構式遺傳演算法

Investigation of Discretizing Real-valued Variables for Discrete Model-Building Genetic Algorithms in Real-valued Optimization

梁哲暐
Che-Wei Liang

指導教授: 于天立 博士
Advisor: Tian-Li Yu, Ph.D.

中華民國 115 年 1 月
January, 2026

doi:10.6342/NTU202600592

摘要

實數最佳化廣泛應用於智慧灌溉、無人機路徑規劃等領域，近年在研究與實務上均受到高度重視。然而，這類問題常同時具有高維度與多峰性，使得求解更加困難。

在高維度情境下，現今的最佳化方法（如 RV-GOMEA）雖展現出優異的搜尋能力，但其模型建構與相依結構的維護成本會隨變數數量增加而快速攀升，進而顯著提高記憶體需求。當問題規模擴大時，記憶體消耗可能呈現急遽成長，使其難以在個人電腦環境中穩定部署，也會對硬體資源受限的邊緣運算場景帶來額外負擔。

爲在維持搜尋能力的同時降低資源需求，本研究採用延伸式精簡基因演算法（ECGA）作爲最佳化後端。相較於 RV-GOMEA，ECGA 具備更緊湊的模型表示，可有效抑制維度成長所帶來的記憶體開銷，進一步提升方法在通用運算環境與邊緣運算場景中的可行性與部署彈性。

在此基礎上，本文進一步提出一種創新的離散化策略，稱爲偏斜多維按需分割（smSoD），針對可分解的高維多峰實數最佳化問題設計，以因應高維度與多峰性並存所造成的搜尋困難。相較於既有的多維按需分割（mSoD），smSoD 透過分析競爭式選擇機制下的樣本分布特性（以球面函數爲分析基礎），將切分點

doi:10.6342/NTU202600592

由隨機選取改爲依據當前樣本分布動態決定。

實驗結果顯示，在兩組基準測試套件——可分解連結問題（decomposable linkage problems）與擴展版 CEC 2014 基準測試（extended version of the CEC 2014 benchmark）——中，無論在已知或未知連結資訊的情境下，smSoD 在高維度設定下展現出最佳的平均排名。在這些情況中，smSoD 的表現優於 mSoD 以及 CEC 2014 競賽優勝演算法 L-SHADE。此外，在僅有 1 GB 記憶體限制的嚴格條件下，smSoD 相較於 RV-GOMEA 同樣達成最佳平均排名，突顯其在計算資源受限的實際應用環境中具備強大的潛力。

關鍵字： 離散化、實數最佳化、基因演算法

doi:10.6342/NTU202600592

Abstract

Real-valued optimization is widely applied in areas such as smart irrigation and UAV path planning, and has attracted significant attention in both research and practice in recent years. However, such problems often exhibit both high dimensionality and multimodality, which makes them particularly difficult to solve.

In high-dimensional settings, modern optimization methods (*e.g.*, Real-valued GOMEA (RV-GOMEA)) can demonstrate strong search capability, but the costs of model construction and dependency-structure maintenance rise rapidly as the number of variables increases, resulting in substantial memory requirements. As the problem scale grows, memory consumption can increase sharply, making stable deployment on personal computers difficult and imposing additional burdens in resource-constrained edge-computing scenarios.

To reduce resource demands while preserving search performance, this study adopts the extended compact genetic algorithm (ECGA) as the optimization backend. Compared with RV-GOMEA, ECGA provides a more compact model representation, which effectively curbs the memory overhead caused by increasing dimensionality, thereby improving feasibility and deployment flexibility in general computing environments and

doi:10.6342/NTU202600592

edge-computing scenarios.

Building on this, we further propose an innovative discretization strategy called skew multidimensional split-on-demand (smSoD) to address the search difficulty arising from the coexistence of high dimensionality and multimodality. Compared with the existing multidimensional split-on-demand (mSoD), smSoD analyzes the sample-distribution characteristics induced by tournament selection (with the spherical function used as the basis of analysis) and replaces random split-point selection with a mechanism that dynamically determines split points according to the current sample distribution.

Experimental results on two benchmark suites—decomposable linkage problems and an extended version of the CEC 2014 benchmark—under both known and unknown linkage settings show that smSoD achieves the best average ranking in high-dimensional settings. In these cases, smSoD ranks ahead of mSoD and L-SHADE (the CEC 2014 competition winner). Furthermore, under a stringent 1 GB memory constraint, smSoD also attains the best average ranking relative to RV-GOMEA, highlighting its strong practical potential in computationally resource-constrained environments.

Keywords: Discretization, Real-valued optimization, Genetic algorithm

doi:10.6342/NTU202600592

Contents

doi:10.6342/NTU202600592

doi:10.6342/NTU202600592

List of Figures

doi:10.6342/NTU202600592

doi:10.6342/NTU202600592

doi:10.6342/NTU202600592

List of Tables

doi:10.6342/NTU202600592

doi:10.6342/NTU202600592

Chapter 1

Introduction

Genetic algorithms (GAs), originally proposed by John Holland [10] and motivated by Darwinian evolution, operate by maintaining a population of candidate solutions. Each candidate is scored through a fitness function, after which better-performing individuals are preferentially chosen to reproduce. New solutions are generated by recombining selected parents, while weaker candidates are removed from the population. Repeating this evaluate–select–recombine loop over many generations generally increases the population's overall fitness. Nevertheless, simple genetic algorithms (SGAs) do not explicitly model or utilize a problem's underlying variable interactions, which limits their effectiveness on problems with strong dependencies among variables, such as trap functions [6].

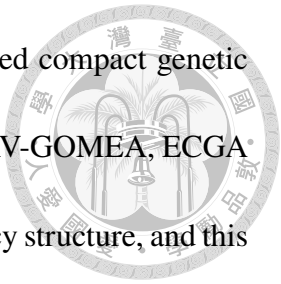
To overcome this limitation, model-building genetic algorithms (MBGAs) infer linkage relationships among genes [12, 15]. By detecting building blocks—groups of mutually dependent variables—MBGAs preserve and recombine these groups as coherent units rather than as independent genes, often leading to faster convergence and better solution quality compared with SGAs.

doi:10.6342/NTU202600592

Among MBGAs, GOMEA [1] is a representative approach for discrete optimization, where linkage information is captured by a linkage tree to guide recombination. For real-valued optimization, Real-valued GOMEA (RV-GOMEA) [2] is the real-valued variant of GOMEA, extending the same core idea to real-valued decision variables. In many high-dimensional tasks, explicitly modeling dependencies can substantially improve search efficiency; however, this benefit comes with a practical cost. As dimensionality increases, the structures and statistics required to learn, store, and update dependency information become progressively heavier, so memory usage can grow quickly with problem scale. In practice, memory often becomes a limiting factor on personal computers, and this limitation is even more pronounced when the optimization algorithm must be deployed on edge devices with tight resource budgets.

This limitation becomes tangible in high-dimensional problems. In smart irrigation [23], discretization over space and time can quickly lead to a massive decision space: a coarse 10×10 spatial grid with 100 time steps yields $10 \times 10 \times 100 = 10{,}000$ spatiotemporal cells, and assigning six coupled variables to each cell results in $10{,}000 \times 6 = 60{,}000$ continuous variables, which can make RV-GOMEA impractical to run on a personal computer due to its rapidly growing memory footprint. A similar issue arises in UAV path planning [17]: representing a path as a K-piece, degree-D Bézier curve in $\mathbb{R}^3$ leads to $K \times (D+1) \times 3$ decision variables; with $K = 30$ and $D = 7$, this already amounts to $30 \times 8 \times 3 = 720$, potentially increasing hardware procurement costs when deployed in edge-computing settings.

doi:10.6342/NTU202600592



Motivated by the above constraint, this thesis adopts the extended compact genetic algorithm (ECGA) [9] as the optimization backend. Compared with RV-GOMEA, ECGA uses a marginal product model (MPM) to represent variable dependency structure, and this model is typically cheaper to maintain as the number of variables grows, making ECGA a practical alternative when memory is the primary bottleneck. This choice improves feasibility for large-scale problems on general-purpose machines and provides greater flexibility for deployment in resource-constrained edge-computing scenarios.

However, ECGA is designed for discrete domains. To enable ECGA to handle continuous-domain problems, split-on-demand (SoD) [3] was proposed as a discretization interface. Yet, because SoD discretizes each component independently, it may ignore real-valued dependencies. To address this limitation, multidimensional split-on-demand (mSoD) [14] was introduced to perform splitting for each group of correlated components, thereby better exploiting inter-component dependencies and improving performance on decomposable multimodal problems [14].

Despite this improvement, high dimensionality and multimodality remain challenging. This thesis proposes a new discretization strategy called skew multidimensional split-on-demand (smSoD). Unlike mSoD, which selects a split point at random, smSoD leverages the sample distribution to choose a split point during discretization. To this end, we analyze how tournament selection shapes the distribution of selected samples on the sphere function and how the split point behaves under this distribution. These observations form the basis of the proposed method.

doi:10.6342/NTU202600592

The remainder of this thesis is organized as follows. Chapter 2 reviews ECGA, mSoD, and the two comparison methods, L-SHADE and RV-GOMEA. Chapter 3 presents the motivation for smSoD and details its integration into ECGA. Chapter 4 introduces the benchmark problems, describes the experimental setup, and discusses the results. Finally, Chapter 5 summarizes the proposed method and findings, and suggests directions for future work.

doi:10.6342/NTU202600592

Chapter 2

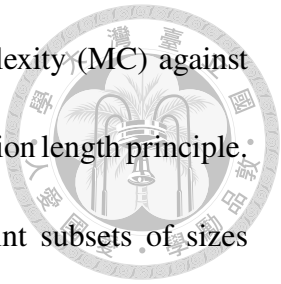
Background

This chapter provides background on ECGA, mSoD, L-SHADE, and RV-GOMEA. We first introduce ECGA, a discrete MBGA that serves as the optimization backend integrated with our discretization framework. We then present mSoD, an adaptive discretization method that interfaces with discrete MBGAs and is used as a key comparative baseline in this work. Finally, we describe L-SHADE and RV-GOMEA, which are used for comparison in this thesis.

2.1 ECGA

ECGA [9], proposed by Harik, replaces traditional crossover and mutation operators with a probabilistic model-based mechanism that generates offspring from a learned distribution. The core idea is that learning a good probability distribution is equivalent to learning linkage. To achieve this, ECGA employs the MPM to represent the population's probability structure.

doi:10.6342/NTU202600592



To determine the optimal model, ECGA balances model complexity (MC) against compressed population complexity (CPC) using the minimum description length principle. Given a population of size N and an MPM consisting of m disjoint subsets of sizes $S_1, \ldots, S_m$, let

$$E(M_i) = -\sum_k p_k \log_2 p_k, \tag{2.1}$$

where p_k denotes the probability of the k-th outcome in subset M_i. The MC and CPC are then defined as

$$\text{MC} = \log_2(N+1) \sum_{i=1}^{m} (2^{S_i} - 1), \tag{2.2}$$

$$\text{CPC} = N \sum_{i=1}^{m} E(M_i). \tag{2.3}$$

By greedily merging subsets to minimize MC + CPC, ECGA constructs a compact yet expressive model. Offspring are subsequently sampled directly from this learned distribution, ensuring the preservation of discovered building blocks. The pseudocode for ECGA is presented in Algorithm 1. Initially developed for binary representations, ECGA has since been extended to integer domains [11] and further integrated with SoD and mSoD frameworks.

doi:10.6342/NTU202600592

Algorithm 1: ECGA

Step 1. Generate an initial population with N individuals at random.

Step 2. Evaluate the fitness of every individual in the population.

Step 3. Select parent candidates using tournament selection.

Step 4. Build the MPM using a greedy search strategy.

Step 5. Draw samples from the MPM to form a new population.

Step 6. End the procedure if the stopping criterion is fulfilled; if not, continue from Step 2.

2.2 mSoD

mSoD [14] is inspired by SoD [3], but it differs in the way variable dependency is handled. In SoD, each component of the decision vector is processed independently. By contrast, mSoD performs splitting over groups of correlated components.

Consider two correlated variables, x_1 and x_2, each defined over the interval $[-100, 100]$, along with a search space consisting of ten points drawn uniformly at random. In both methods, any region that contains at least five points is further partitioned. SoD treats the two dimensions as independent and discretizes them separately, while mSoD uses prior linkage knowledge to discretize the variables together. As shown in Figure 2.1, SoD applies one-dimensional discretization per axis and repeatedly subdivides any interval whose point count exceeds the threshold. By comparison, Figure 2.2 demonstrates that mSoD carries out discretization in multiple dimensions simultaneously and similarly continues splitting whenever a region contains too many points.

At each iteration, mSoD samples a split location uniformly from the current region R and partitions R into 2^d cells. A cell containing at least $N \cdot \gamma$ individuals is recursively split,

doi:10.6342/NTU202600592

where γ is the splitting threshold and N is the population size. As evolution proceeds, the threshold is reduced by $\gamma \leftarrow \varepsilon \cdot \gamma$. By explicitly leveraging variable interactions, mSoD can be more effective than SoD on problems where linkage information is available [14]. The complete procedure is summarized in Algorithm 2 and Algorithm 3.

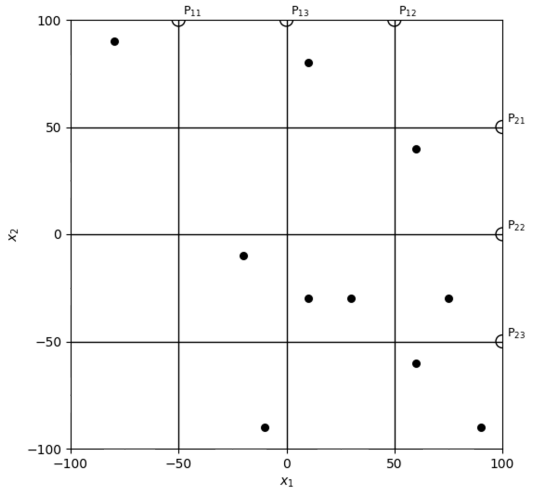


Figure 2.1: An example of how SoD may partition regions in a search space containing ten uniformly sampled points.

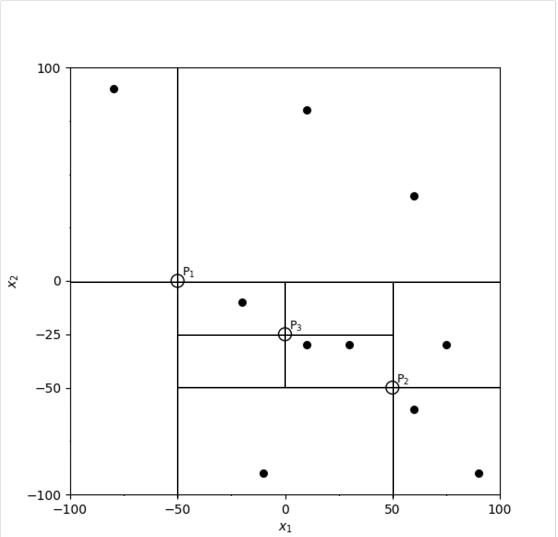


Figure 2.2: An example of how mSoD may partition regions in a search space containing ten uniformly sampled points.

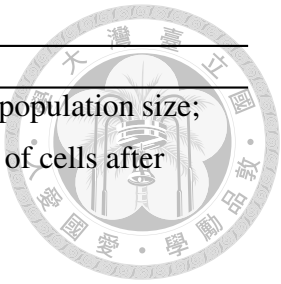
Algorithm 2: mSoD

Inputs: R: region to be processed; γ: splitting threshold; N: population size; ε: decay factor for γ;

1 mSplit(R, γ, N)

2 $\gamma \leftarrow \varepsilon \cdot \gamma$

doi:10.6342/NTU202600592



Algorithm 3: mSplit

Inputs: R: region under consideration; γ: splitting threshold; N: population size;

Items: d: dimensionality of R; C: sampled split location; $\mathscr{B}$: set of cells after partitioning; B_k: the k-th cell in $\mathscr{B}$;

```
Draw a split location C uniformly from R
𝓑 ← Partition R into 2^d cells using C
foreach B_k ∈ 𝓑 do
    c_k ← count individuals in B_k
    if c_k ≥ N · γ then
        mSplit(B_k, γ, N)
    else if c_k > 0 then
        Assign an identifier (code) to B_k
    else
        Remove B_k
```

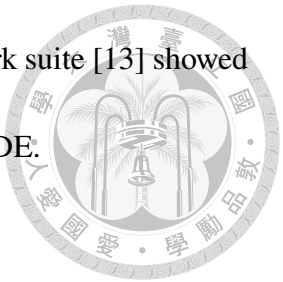
2.3 L-SHADE

Differential evolution (DE) was proposed to solve real-valued optimization problems [19]. There are three major control parameters in DE: population size N, scaling factor F, and crossover rate CR. The settings of these parameters significantly influence optimization performance. To address this issue for F and CR, success-history-based adaptive differential evolution (SHADE) [20] was proposed; it adaptively adjusts F and CR based on historically successful parameter values.

While SHADE automatically adjusts F and CR, the population size N remains constant throughout the search. To further enhance the performance of SHADE, Tanabe and Fukunaga [21] proposed L-SHADE by incorporating linear population size reduction (LPSR), which continuously reduces the population size according to a linear function of

doi:10.6342/NTU202600592

the number of fitness evaluations. Results on the CEC 2014 benchmark suite [13] showed that L-SHADE significantly improves upon the performance of SHADE.



2.4 RV-GOMEA

RV-GOMEA [2], inspired by GOMEA [1], is a state-of-the-art real-valued model-based genetic algorithm (MBGA) [18] designed for continuous optimization problems. GOMEA employs a family of subsets (FOS) to represent dependencies among problem variables. A FOS consists of multiple subsets, where each subset corresponds to a group of interdependent variables. In the linkage-tree FOS, the structure is initialized with subsets containing a single variable and iteratively expanded by merging the two most mutually dependent subsets. The resulting linkage tree contains $2d-1$ subsets, where d is the number of decision variables.

RV-GOMEA selects a subset S of promising solutions and, for each subset F_j in the FOS, estimates a normal distribution $\mathcal{N}(\hat{\mu}_{F_j}, \hat{\Sigma}_{F_j})$ from S. During variation, the subsets F_j are processed in a random order, and these distributions are used to sample partial vectors over the variables in F_j. Elitist solutions are copied unchanged, and for each non-elitist solution P_i in the population, a sampled partial vector y_{F_j} is inserted into P_i. If the resulting solution improves the fitness, the change is kept; otherwise, the modification is kept with probability $p_{\text{accept}} = 0.05$; if not kept, the solution P_i is restored.

doi:10.6342/NTU202600592

Chapter 3

Methodology

In this chapter, we introduce smSoD, a method that leverages the sample distribution to determine the split point during the discretization process. We begin by outlining the motivation behind smSoD, then provide a detailed description of its procedure, and conclude by explaining how smSoD can be integrated into ECGA. The source code is publicly available at `https://github.com/wacly/smSoD`

3.1 Motivation

3.1.1 Effect of Split-Point Choice on Optimization Performance

We hypothesize that an unsuitable choice of split point in the discretization stage can weaken effective exploration. To test this hypothesis, we performed experiments on a toy example and on the CEC 2014 benchmark suite [13]; the results align with our hypothesis.

For the toy example, we use the one-dimensional sphere function as the minimization

doi:10.6342/NTU202600592

objective:

$$s(x) = x^2, \quad x \in [-1, 1]. \tag{3.1}$$

Assume that ten points are sampled uniformly from the interval $[-1,1]$, as shown in Figure 3.1. If we choose the midpoint as the split location, $q_1 = 0$, the interval is divided into two subintervals, $[-1,0)$ and $[0,1]$. With this partition, six sampled points fall in the left subinterval and four fall in the right. Based on this split and the corresponding sample distribution, we then construct a new probability density function as follows:

$$p_{q_1}(x) = \begin{cases} \dfrac{\frac{6}{10}}{0-(-1)} = \frac{3}{5}, & x \in [-1, 0), \\ \dfrac{\frac{4}{10}}{1-0} = \frac{2}{5}, & x \in [0, 1]. \end{cases} \tag{3.2}$$

Under this piecewise-uniform distribution, the expected fitness is approximately

$$\mathbb{E}[s(X)] \approx 0.333, \tag{3.3}$$

where X is a random variable with density $p_{q_1}(x)$.

By contrast, if we select the split point at $q_2 = \frac{2}{3}$, nine of the sampled points fall within the left subinterval $\left[-1, \frac{2}{3}\right)$ and one within the right subinterval $\left[\frac{2}{3}, 1\right]$, as illustrated in Figure 3.1. Based on this split point and the corresponding sample distribution, we

doi:10.6342/NTU202600592

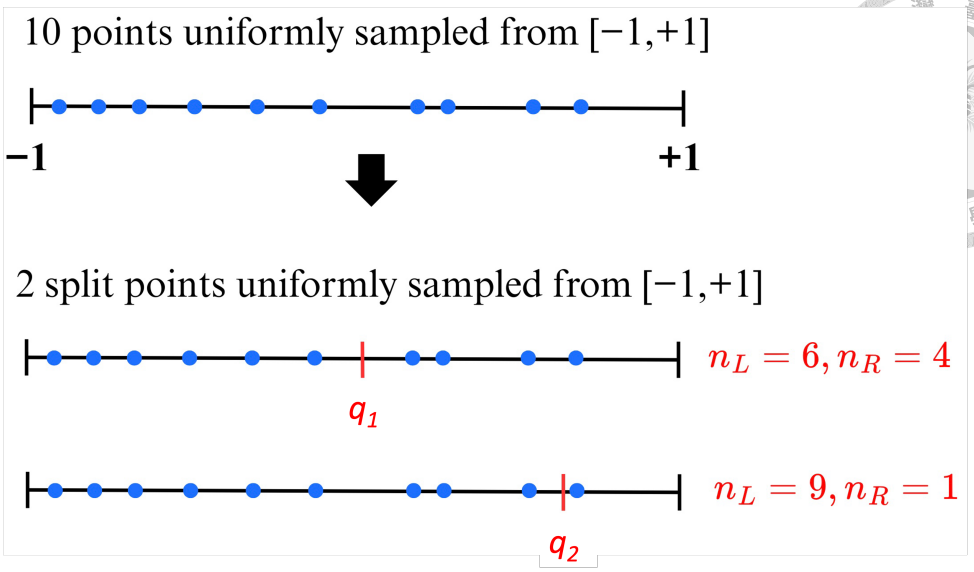


Figure 3.1: Illustration of ten uniformly sampled points over the interval $[-1, 1]$, with split points at $q_1 = 0$ and $q_2 = \frac{2}{3}$. Here, n_L and n_R denote the numbers of sampled points in the left and right subintervals, respectively. Compared with q_1, selecting the split point q_2 yields a lower expected fitness.

construct a new probability density function as follows:

$$p_{q_2}(x) = \begin{cases} \dfrac{\frac{9}{10}}{\frac{2}{3}-(-1)} = \frac{27}{50}, & x \in \left[-1, \frac{2}{3}\right), \\ \dfrac{\frac{1}{10}}{1-\frac{2}{3}} = \frac{3}{10}, & x \in \left[\frac{2}{3}, 1\right]. \end{cases} \tag{3.4}$$

Under this piecewise-uniform distribution, the expected fitness is approximately

$$\mathbb{E}[s(Y)] \approx 0.304, \tag{3.5}$$

where Y is a random variable with density $p_{q_2}(x)$. Therefore, selecting $q_2 = \frac{2}{3}$ as the split point yields a better expected fitness than selecting the midpoint $q_1 = 0$.

To develop a principled criterion for selecting a split point, we revisit the sphere

doi:10.6342/NTU202600592

function:

$$s(x) = x^2, \qquad x \in [-1, 1]. \tag{3.6}$$

The sphere function [5] has long been studied in optimization due to its simplicity, convexity, differentiability, and scalability. With sufficiently fine resolution, an objective function can be locally approximated by its second-order Taylor expansion, and because an interior local extremum requires a nonzero second-order term, the sphere function is the simplest form and thus a natural basis for analysis.

Next, we adopt binary tournament selection. In addition to being the selection operator used in ECGA, tournament selection retains the noise robustness of ranking-based selection operators [7]. It is also computationally efficient, as it avoids ranking the entire population. Under this scheme, the probability density function on $[-1, 1]$ is

$$f(x) = 1 - |x|, \quad x \in [-1, 1], \tag{3.7}$$

and the corresponding cumulative distribution function is

$$F(x) \;=\; \int_{-1}^{x} \big(1 - |t|\big)\, dt. \tag{3.8}$$

After splitting the interval at a point $q \in [-1, 1]$, the probability that a sample falls within $[-1, q)$ is $F(q)$, and the probability that it falls within $[q, 1]$ is $1 - F(q)$. Based on this split point and the corresponding sample distribution, we construct a new probability

doi:10.6342/NTU202600592

density function as follows:

$$p_q(x) = \begin{cases} \dfrac{F(q)}{q+1}, & x \in [-1, q), \\ \dfrac{1-F(q)}{1-q}, & x \in [q, 1]. \end{cases} \tag{3.9}$$

Under this scheme, the expected fitness is given by

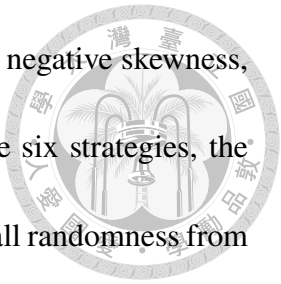
$$\mathbb{E}[s(X)] = \frac{F(q)}{q+1}\int_{-1}^{q} x^2\,dx + \frac{1-F(q)}{1-q}\int_{q}^{1} x^2\,dx, \tag{3.10}$$

where X is a random variable with density $p_q(x)$. We evaluate the quality of a candidate density by its expected fitness $\mathbb{E}[s(X)]$, which represents the mean fitness of offspring sampled from p_q. We use an expectation (average-case) criterion rather than a minimum-based criterion because offspring generation is inherently stochastic, so the mean fitness better reflects typical progress, whereas a minimum objective would be dominated by low-probability tail samples and thus be unstable.

In this example, the expected fitness is minimized at $q = \pm\frac{2}{3}$. On the interval $[-1, 1]$, measured from the left endpoint -1, these positions correspond to $q = -\frac{2}{3}$ at $\frac{1}{6}$ of the interval length and $q = \frac{2}{3}$ at $\frac{5}{6}$ of the interval length.

To further assess the generality of our findings, we conducted experiments on 30 test functions from the CEC 2014 benchmark suite, each in 10 dimensions, using ECGA as the optimization backend. We compared six discretization strategies for choosing the split point (Table 3.1). The first strategy uses a fixed split located one-sixth of the interval

doi:10.6342/NTU202600592



length from the left endpoint. Under our experimental assumption of negative skewness, this split location is applied consistently across all runs. Among the six strategies, the fixed-split approach performed the worst, likely because it eliminated all randomness from the discretization process.

To restore variability, we evaluated five stochastic alternatives:

1. **Big window:** The split point is drawn uniformly from a window centered at the one-sixth position, with total width $\frac{1}{3}$ (*i.e.*, radius $\frac{1}{6}$).

2. **Small window:** The split point is drawn uniformly from a narrower window centered at the one-sixth position, with total width $\frac{1}{6}$ (*i.e.*, radius $\frac{1}{12}$).

3. **Beta distribution:** The split point is sampled from a Beta distribution whose mean is located at one-sixth of the interval.

4. **Uniform distribution:** The split point is drawn uniformly from the entire interval.

5. **Skew selection:** The split point is determined using an order-statistics-inspired procedure, as described in Section 3.2.

Table 3.1: Average ranking across 30 CEC 2014 benchmark functions (10 dimensions) for six discretization methods. The skew selection achieves the best overall performance.

Method	Average Ranking
Fixed split	4.300
Big window	3.067
Small window	3.467
Beta distribution	3.033
Uniform distribution	4.200
Skew selection	**2.933**

doi:10.6342/NTU202600592

3.1.2 Generalization to Higher Dimensions

In this subsection, we extend the analysis in Section 3.1.1, which focuses on the one-dimensional sphere function, to higher-dimensional settings. We first introduce the notation used throughout this subsection (Table 3.2) and then organize the discussion into three parts: (1) necessary conditions for the optimal split point, (2) probability density induced by binary tournament selection, and (3) marginal analysis and a central limit theorem–based approximation of the optimal split point in higher dimensions.

Table 3.2: Important symbols and their meanings used in this paper.

Symbol	Description
n	Number of dimensions.
Ω	Sample space $[-1,1]^n$ (candidate solution set).
$g(\boldsymbol{x})$	Probability density function (PDF) of $\boldsymbol{x}$ after binary tournament selection, where $\boldsymbol{x} = (x_1, \ldots, x_n) \in \Omega$.
$\{R_{\boldsymbol{k}}(\boldsymbol{q}) : \boldsymbol{k} \in \{0,1\}^n\}$	Partition of Ω induced by $\boldsymbol{q} = (q_1, \ldots, q_n) \in \Omega$ via the mSoD procedure.
$R_{\boldsymbol{k}}(\boldsymbol{q})$	For any $\boldsymbol{k} = (k_1, \ldots, k_n) \in \{0,1\}^n$, the region $R_{\boldsymbol{k}}(\boldsymbol{q}) = I_{\boldsymbol{k},1} \times I_{\boldsymbol{k},2} \times \cdots \times I_{\boldsymbol{k},n}$.
$I_{\boldsymbol{k},i}$	$I_{\boldsymbol{k},i} = \begin{cases} [-1, q_i), & k_i = 0, \\ [q_i, 1], & k_i = 1. \end{cases}$
$\lvert I_{\boldsymbol{k},i}\rvert$	$\lvert I_{\boldsymbol{k},i}\rvert = \begin{cases} q_i + 1, & k_i = 0, \\ 1 - q_i, & k_i = 1. \end{cases}$
$\lvert R_{\boldsymbol{k}}(\boldsymbol{q})\rvert$	$\prod_{i=1}^{n} \lvert I_{\boldsymbol{k},i}\rvert$.
$\Pr(R_{\boldsymbol{k}}(\boldsymbol{q}))$	Probability of the event $R_{\boldsymbol{k}}(\boldsymbol{q})$ under g, *i.e.*, $\int_{R_{\boldsymbol{k}}(\boldsymbol{q})} g(\boldsymbol{x})\, \mathrm{d}\boldsymbol{x}$.
$s(\boldsymbol{x})$	Sphere function defined by $s(\boldsymbol{x}) = \sum_{i=1}^{n} x_i^2$.
$\boldsymbol{X}$	A random vector $\boldsymbol{X} = (X_1, \ldots, X_n)$ taking values in Ω.
$s(\boldsymbol{X})$	Random variable defined by $s(\boldsymbol{X})(\omega) = \sum_{i=1}^{n} X_i(\omega)^2$, for $\omega \in \Omega$.

doi:10.6342/NTU202600592

Necessary Conditions for the Optimal Split Point

In this subsection, we derive a necessary condition that any optimal split point should satisfy. According to the mSoD framework described in Section 2.2, once a split point $\boldsymbol{q} \in \Omega$ is selected, the search space is partitioned into subregions $\{R_{\boldsymbol{k}}(\boldsymbol{q}) : \boldsymbol{k} \in \{0,1\}^n\}$. The resulting probability density function is piecewise constant and can be written as

$$p_{\boldsymbol{q}}(\boldsymbol{x}) = \frac{\Pr(R_{\boldsymbol{k}}(\boldsymbol{q}))}{|R_{\boldsymbol{k}}(\boldsymbol{q})|} \quad \text{for } \boldsymbol{x} \in R_{\boldsymbol{k}}(\boldsymbol{q}),\ \boldsymbol{k} \in \{0,1\}^n. \tag{3.11}$$

Our goal is to choose a split point $\boldsymbol{q}$ that minimizes the expected value of s, which can be written as

$$\mathbb{E}\big[s(\boldsymbol{X})\big] = \sum_{\boldsymbol{k}} \frac{\Pr\big(R_{\boldsymbol{k}}(\boldsymbol{q})\big)}{|R_{\boldsymbol{k}}(\boldsymbol{q})|} \int_{R_{\boldsymbol{k}}(\boldsymbol{q})} s(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} = \sum_{i=1}^{n} \sum_{\boldsymbol{k}} \Pr\big(R_{\boldsymbol{k}}(\boldsymbol{q})\big)\, u_{\boldsymbol{k},i}(q_i), \tag{3.12}$$

where $\boldsymbol{X}$ is a random vector with density $p_{\boldsymbol{q}}(\boldsymbol{x})$, and $u_{\boldsymbol{k},i}(q_i)$ is defined as

$$u_{\boldsymbol{k},i}(q_i) = \frac{1}{|R_{\boldsymbol{k}}(\boldsymbol{q})|} \int_{R_{\boldsymbol{k}}(\boldsymbol{q})} x_i^2\,\mathrm{d}\boldsymbol{x} = \frac{1}{\prod_{j=1}^{n} |I_{k,j}|} \left(\prod_{j \neq i} |I_{k,j}| \right) \int_{I_{k,i}} x_i^2\,\mathrm{d}x_i = \frac{1}{|I_{k,i}|} \int_{I_{k,i}} x_i^2\,\mathrm{d}x_i. \tag{3.13}$$

Next, we decompose $u_{\boldsymbol{k},i}(q_i)$ into two cases:

$$u_{\boldsymbol{k},i}^{-}(q_i) = \frac{1}{1+q_i} \int_{-1}^{q_i} x^2\,\mathrm{d}x = \frac{q_i^2 - q_i + 1}{3}, \qquad u_{\boldsymbol{k},i}^{+}(q_i) = \frac{1}{1-q_i} \int_{q_i}^{1} x^2\,\mathrm{d}x = \frac{q_i^2 + q_i + 1}{3}. \tag{3.14}$$

doi:10.6342/NTU202600592

Substituting the above expressions, we obtain

$$\sum_{\boldsymbol{k}} \Pr\big(R_{\boldsymbol{k}}(\boldsymbol{q})\big)\, u_{\boldsymbol{k},i}(q_i) = u^{-}_{\boldsymbol{k},i}(q_i)\, F(q_i) + u^{+}_{\boldsymbol{k},i}(q_i)\big(1-F(q_i)\big) = \frac{q_i^2+q_i+1}{3} - \frac{2q_i}{3}F(q_i), \tag{3.15}$$

where

$$F(q_i) = \sum_{\substack{\boldsymbol{k}\in\{0,1\}^n \\ k_i=0}} \Pr\big(R_{\boldsymbol{k}}(\boldsymbol{q})\big). \tag{3.16}$$

If F is differentiable and we denote $f = F'$, then the critical points q_i^* satisfy

$$\frac{\mathrm{d}}{\mathrm{d}q_i}\left[\frac{q_i^2+q_i+1}{3} - \frac{2q_i}{3}F(q_i)\right]_{q_i=q_i^*} = 0 \quad\Longleftrightarrow\quad F(q_i^*) + q_i^* f(q_i^*) = \frac{1+2q_i^*}{2}. \tag{3.17}$$

We now consider the case $n = 1$, for which

$$F(q_1) = \begin{cases} \dfrac{1}{2} + q_1 + \dfrac{q_1^2}{2}, & q_1 \le 0, \\[2ex] \dfrac{1}{2} + q_1 - \dfrac{q_1^2}{2}, & q_1 > 0, \end{cases} \qquad f(q_1) = 1 - |q_1|. \tag{3.18}$$

In this case, the solutions to the optimality condition

$$F(q_1) + q_1 f(q_1) = \frac{1+2q_1}{2} \tag{3.19}$$

are $q_1 \in \{-\frac{2}{3}, 0, \frac{2}{3}\}$, which is consistent with the result obtained in Section 3.1.1.

doi:10.6342/NTU202600592

Probability Density Induced by Binary Tournament Selection

In this subsection, we derive the form of g. Specifically,

$$g(\boldsymbol{y}) = \frac{2}{2^n} \Pr\big(s(\boldsymbol{y}) \leq s(\boldsymbol{X})\big), \tag{3.20}$$

where the notation follows Table 3.2, $\boldsymbol{y} \in \Omega$, and $\boldsymbol{X} \sim \mathrm{Unif}(\Omega)$. To verify that g integrates to 1 over Ω, introduce the auxiliary function

$$g'(\boldsymbol{y}) = \frac{1}{2^n}\Big(\Pr\big(s(\boldsymbol{y}) \leq s(\boldsymbol{X})\big) + \Pr\big(s(\boldsymbol{y}) > s(\boldsymbol{X})\big)\Big) = \frac{1}{2^n}, \tag{3.21}$$

where the two events $\{s(\boldsymbol{y}) \leq s(\boldsymbol{X})\}$ and $\{s(\boldsymbol{y}) > s(\boldsymbol{X})\}$ are mutually exclusive and collectively exhaustive; therefore, their probabilities sum to one. Hence,

$$\int_{\Omega} g'(\boldsymbol{y})\,\mathrm{d}\boldsymbol{y} = 1. \tag{3.22}$$

Moreover, observe that

$$\begin{aligned}\int_{\Omega}\big(g(\boldsymbol{y}) - g'(\boldsymbol{y})\big)\,\mathrm{d}\boldsymbol{y} &= \int_{\Omega} \frac{1}{2^n}\big(\Pr\big(s(\boldsymbol{y}) \leq s(\boldsymbol{X})\big) - \Pr\big(s(\boldsymbol{y}) > s(\boldsymbol{X})\big)\big)\,\mathrm{d}\boldsymbol{y} \\ &= \Pr\big(s(\boldsymbol{Y}) \leq s(\boldsymbol{X})\big) - \Pr\big(s(\boldsymbol{Y}) > s(\boldsymbol{X})\big) \\ &= 0,\end{aligned} \tag{3.23}$$

where $\boldsymbol{X}, \boldsymbol{Y} \overset{\text{i.i.d.}}{\sim} \mathrm{Unif}(\Omega)$. This confirms that the proposed form of g is properly normalized

doi:10.6342/NTU202600592

over Ω.

To further validate the normalization of g, we estimated $\int_\Omega g$ via Monte Carlo simulation for several dimensions $n \in \{1, 2, 5, 10, 1000\}$. As shown in Table 3.3, all estimated values are close to 1, as expected.

Table 3.3: Monte Carlo simulation results for $\int_\Omega g$ across different dimensions n.

n	Estimated $\int_\Omega g$
1	1.0000
2	0.9996
5	0.9995
10	1.0002
1000	0.9993

Marginal Analysis and a Central Limit Theorem–Based Approximation of the Optimal Split Point in Higher Dimensions

In this subsection, we build on the results of the previous two subsections and extend the optimal split-point selection rule to the higher-dimensional setting. Throughout, we continue to use the notation introduced in Table 3.2. In particular, we define

$$\boldsymbol{X} = (X_1, \ldots, X_n), \qquad X_1, \ldots, X_n \overset{\text{i.i.d.}}{\sim} \text{Unif}([-1, 1]), \qquad \boldsymbol{X}_{2:n} = (X_2, \ldots, X_n), \tag{3.24}$$

and introduce the following quantities:

$$\boldsymbol{X} \perp\!\!\!\perp \boldsymbol{Y}, \qquad U = \sum_{i=2}^{n} X_i^2, \qquad R = s(\boldsymbol{Y}). \tag{3.25}$$

To study the optimal split point in high dimensions, we focus on the marginal density

doi:10.6342/NTU202600592

of the first coordinate under the joint density g (by symmetry, any coordinate would yield the same result). Define

$$\begin{aligned} m(x_1) &= \int_{[-1,1]^{n-1}} g(x_1, x_2, \ldots, x_n)\, \mathrm{d}x_2 \cdots \mathrm{d}x_n \\ &= \int_{[-1,1]^{n-1}} \frac{1}{2^{n-1}} \Pr\Big(s(\boldsymbol{Y}) \geq x_1^2 + \sum_{i=2}^{n} x_i^2 \Big)\, \mathrm{d}x_2 \cdots \mathrm{d}x_n \\ &= \Pr\big(R \geq x_1^2 + U\big). \end{aligned} \tag{3.26}$$

By the central limit theorem, both R and U are approximately normal for large n. Let $Z \sim \mathrm{Unif}([-1,1])$. Then

$$\mu = \mathbb{E}[Z^2] = \frac{1}{3}, \qquad \sigma^2 = \mathrm{Var}(Z^2) = \frac{4}{45}. \tag{3.27}$$

Since $\boldsymbol{X} \perp\!\!\!\perp \boldsymbol{Y}$, we have

$$\mathbb{E}[R-U] = \mathbb{E}[R] - \mathbb{E}[U] = n\mu - (n-1)\mu = \mu, \tag{3.28}$$

and, by independence,

$$\mathrm{Var}(R-U) = \mathrm{Var}(R) + \mathrm{Var}(U) = n\sigma^2 + (n-1)\sigma^2 = (2n-1)\sigma^2. \tag{3.29}$$

Hence,

$$R - U \approx \mathcal{N}\big(\mu, (2n-1)\sigma^2\big), \tag{3.30}$$

doi:10.6342/NTU202600592

and therefore

$$m(x_1) = \Pr(R - U \geq x_1^2) \approx 1 - \Phi\left(\frac{x_1^2 - \mu}{\sqrt{(2n-1)\sigma^2}}\right). \tag{3.31}$$

Using the approximation in Abernathy (1988) [4],

$$m(x_1) \approx \frac{1}{2} - \frac{1}{\sqrt{2\pi}} \frac{x_1^2 - \frac{1}{3}}{\sqrt{2n-1}\,\sigma}. \tag{3.32}$$

Let $M(x_1)$ denote the integral of $m(x)$ from -1 to x_1:

$$M(x_1) = \int_{-1}^{x_1} m(x)\,dx \;\approx\; \frac{x_1 + 1}{2} - \frac{x_1^3 - x_1}{3\sqrt{2\pi}\sqrt{2n-1}\,\sigma}. \tag{3.33}$$

We then solve the equation

$$M(x_1) + x_1\, m(x_1) \;=\; \frac{1 + 2x_1}{2}. \tag{3.34}$$

The values

$$x_1 \in \left\{-\tfrac{1}{\sqrt{2}}, 0, \tfrac{1}{\sqrt{2}}\right\} \tag{3.35}$$

satisfy the above equation.

To validate the above results, we conduct Monte Carlo simulations for $n \in \{1, 2, 5, 10, 1000\}$ and estimate the corresponding critical points. We also compute the theoretical critical points implied by the approximation derived above. As shown in Table 3.4, the Monte Carlo estimates closely match the theoretical values across all dimensions considered, and

doi:10.6342/NTU202600592

the critical point approaches $1/\sqrt{2}$ as n increases.

Table 3.4: Estimated and theoretical critical points across dimensions

n	Monte Carlo mean of critical point	Theoretical critical point
1	0.66848	0.66667
2	0.69253	0.69206
5	0.70250	0.70243
10	0.70292	0.70494
1000	0.70704	0.70711

3.2 smSoD

In mSoD, the split point is sampled uniformly from the current region to be divided, which induces a partition of the d-dimensional domain into 2^d child subregions. In smSoD, however, we adopt a candidate-driven strategy: multiple potential split points are generated first, and the essential difference is the criterion used to select the final split location. As noted in Section 3.1, empirically good splits tend to occur around the one-sixth and five-sixths positions of an interval. Motivated by this pattern, we leverage results from order statistics to determine an appropriate number of candidate split points to draw.

Let

$$X_1, \ldots, X_n \sim \text{Uniform}(0,1), \qquad X_{(1)} \leq \cdots \leq X_{(n)}, \tag{3.36}$$

where $U_{(i)}$ denotes the ith smallest sample (the ith order statistic). The expected minimum

doi:10.6342/NTU202600592

and maximum satisfy

$$\mathbb{E}[X_{(1)}] = \frac{1}{n+1}, \qquad \mathbb{E}[X_{(n)}] = \frac{n}{n+1}. \tag{3.37}$$

Setting these expectations equal to $\frac{1}{6}$ and $\frac{5}{6}$, respectively, yields $n = 5$. Based on this reasoning, we introduce the following split-point selection method, referred to as skew selection. Our design is motivated by a one-dimensional order-statistic analysis; in higher dimensions, we adopt a fixed candidate set size, which we found to be sufficient for stable performance in our experiments. We note, however, that the optimal number of candidate split points can depend on the problem dimensionality and the selection size. In addition, we explicitly account for the sample distribution: in black-box optimization, the objective function is unknown, so we rely on the information provided by the selection operator to redesign the discretization strategy accordingly.

1. **Sampling split candidates.** Draw five candidate split locations $\{S_1, \ldots, S_5\}$ independently from a uniform distribution over the target region.

2. **Subregion construction.** Given a candidate S_i, split the target region at S_i along each coordinate axis, yielding 2^d subregions $\{R_{i,1}, \ldots, R_{i,2^d}\}$. Denote by $n_{i,j}$ the number of individuals falling into subregion $R_{i,j}$.

3. **Choosing the split location.** For each candidate S_i, compute the size of its densest subregion:

$$N_i = \max_{1 \le j \le 2^d} n_{i,j}.$$

doi:10.6342/NTU202600592

We then select the candidate that maximizes this quantity:

$$P^* = \arg\max_{1 \le i \le 5} N_i, \qquad \text{and set the split point to } S_{P^*}.$$

Figure 3.2 illustrates the one-dimensional case with $N = 10$ sampled points. The five candidates produce maximum subinterval occupancies of $[8, 5, 9, 7, 6]$; therefore, the third candidate (S_3) is chosen as the split location.

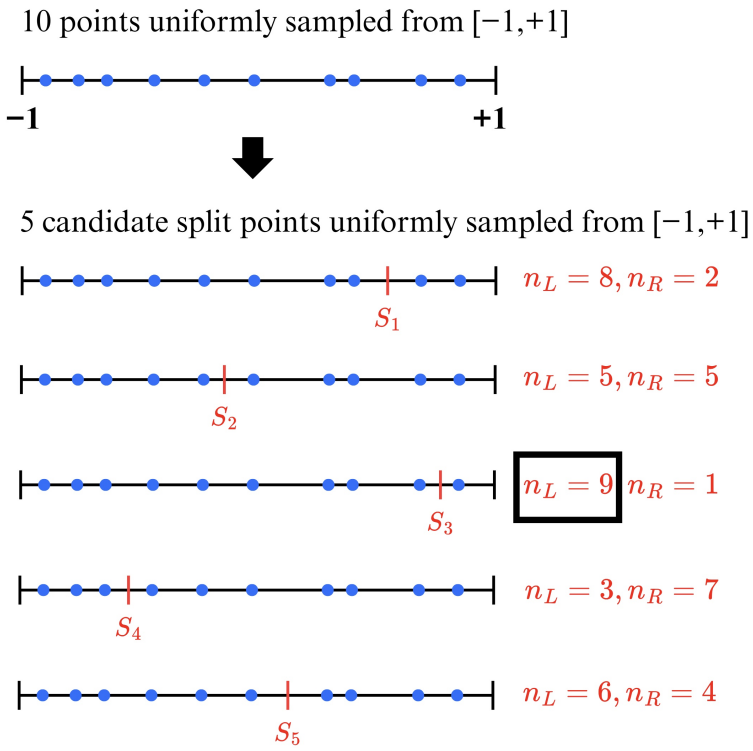
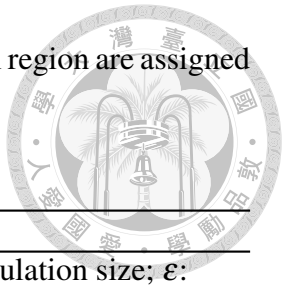


Figure 3.2: A one-dimensional illustration on $[-1, 1]$ with ten uniformly sampled points and five candidate split locations $S_1, \ldots, S_5$. The quantities n_L and n_R denote the numbers of points in the left and right subintervals, respectively. Under skew selection, S_3 is chosen because it yields the largest count in the densest resulting subregion.

The discussion above focuses on split-location selection. Overall, Algorithm 4 calls mSplit with the current resolution parameter γ and then updates it as $\gamma \leftarrow \varepsilon \cdot \gamma$. Algorithm 5 recursively partitions R whenever a region contains at least $N \cdot \gamma$ individuals, and terminates

doi:10.6342/NTU202600592

when all regions fall below this threshold. Individuals in the same final region are assigned the same discrete code, while empty regions are discarded.



Algorithm 4: smSoD

Inputs: R: region to be processed; γ: splitting threshold; N: population size; ε: decay factor for γ;

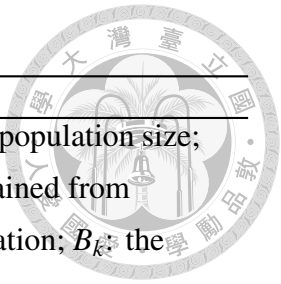
1 mSplit(R, γ, N)

2 $\gamma \leftarrow \varepsilon \cdot \gamma$

3.3 Integration of smSoD into ECGA

Similar to SoD and mSoD, smSoD only performs the discretization of continuous variables and therefore relies on a backend optimization algorithm to solve the full problem. In this study, we employ ECGA as the backend optimizer. The pseudocode of smSoD + ECGA is given below:

doi:10.6342/NTU202600592



Algorithm 5: mSplit

Inputs: R: region under consideration; γ: splitting threshold; N: population size;

Items: C_i: the i-th candidate split location; $B_{i,j}$: the j-th cell obtained from candidate C_i; d: dimensionality of R; $\hat{C}$: selected split location; B_k: the k-th cell after splitting with $\hat{C}$;

$\{C_i\}_{i=1}^{5} \leftarrow$ Generate 5 candidate split locations within R
for $i \leftarrow 1$ **to** 5 **do**
    $\mathscr{B}_i \leftarrow$ Partition R into 2^d cells using C_i
    foreach $B_{i,j} \in \mathscr{B}_i$ **do**
        $c_{i,j} \leftarrow$ Count individuals falling in $B_{i,j}$
    $M_i \leftarrow \max_{1 \le j \le 2^d} c_{i,j}$
$q \leftarrow \operatorname{argmax}_{1 \le i \le 5} M_i$
$\hat{C} \leftarrow C_q$
$\mathscr{B} \leftarrow$ Partition R into 2^d cells using $\hat{C}$
foreach $B_k \in \mathscr{B}$ **do**
    $c_k \leftarrow$ Count individuals in B_k
    if $c_k \geq N \cdot \gamma$ **then**
        mSplit(B_k, γ, N)
    else if $c_k > 0$ **then**
        Attach an identifier to B_k
    else
        Remove B_k

Algorithm 6: smSoD + ECGA

Step 1. Generate an initial population with N individuals at random.

Step 2. Evaluate the fitness of every individual in the population.

Step 3. Select parent candidates using tournament selection.

Step 4. Encode the selected candidates through smSoD.

Step 5. Build the MPM using a greedy search strategy.

Step 6. Draw samples from the MPM to form a new population.

Step 7. End the procedure if the stopping criterion is fulfilled; if not, continue from Step 2.

doi:10.6342/NTU202600592

Chapter 4

Experiments

In this chapter, we begin by introducing the benchmark suites and outlining their main characteristics. The benchmarks consist of two parts: a collection of decomposable linkage problems [12], and an extended CEC 2014 benchmark [11] in which extra subproblems are added. Based on these benchmarks, we examine three experimental settings: (1) performance with linkage information provided, (2) performance without linkage information, and (3) performance under memory limitations. We then describe the parameter settings and implementation details of each method, followed by the experimental results with a focus on comparative evaluation. Lastly, we assess the additional computational cost incurred by smSoD relative to mSoD.

4.1 Test Problems

We assess the algorithms using two test suites: decomposable linkage problems and an extended version of the CEC 2014 benchmark that includes additional subproblems.

doi:10.6342/NTU202600592

Both benchmark suites feature decomposable problem structures and strongly multimodal fitness landscapes. For each problem, the n-dimensional decision vector $\mathbf{x} \in \mathbb{R}^n$ is divided into m disjoint subvectors of equal size k:

$$\mathbf{x} = [\mathbf{x}_1, \ldots, \mathbf{x}_m], \quad \mathbf{x}_i \in \mathbb{R}^k.$$

Each subvector $\mathbf{x}_i$ corresponds to an independent subproblem with objective function $f_{\text{sub}}(k, \mathbf{x}_i, i)$, whose global minimum is known. Here, k is the subproblem dimension, $\mathbf{x}_i$ is the associated subvector, and i denotes the subvector index. The overall objective is then formed by aggregating the subproblem objectives, resulting in a composite function that contains a large number of local optima. This formulation yields a challenging optimization landscape that is highly multimodal, with independent subproblems.

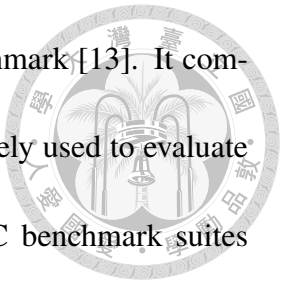
The first benchmark suite was originally proposed to evaluate mSoD [14] on decomposable problems with multi-modal landscapes. The i-th subproblem is defined as

$$f_{\text{sub}}(k, \mathbf{x}_i, i) \;=\; \big((\mathbf{x}_i)_1 - c_i\big)^2 \;+\; \sum_{j=1}^{k-1} \big((\mathbf{x}_i)_{j+1} - (\mathbf{x}_i)_j\big)^2,$$

where c_i is a subproblem-specific constant uniquely associated with index i. The full objective is then formed by summing the contributions of all subproblems:

$$f(m, k, \mathbf{x}) \;=\; \sum_{i=1}^{m} f_{\text{sub}}(k, \mathbf{x}_i, i).$$

doi:10.6342/NTU202600592



The other benchmark suite is derived from the CEC 2014 benchmark [13]. It comprises 30 test problems with diverse characteristics and has been widely used to evaluate the performance of optimization algorithms. Although newer CEC benchmark suites have been released, CEC 2014 remains a representative and commonly adopted test set [8, 16]. In this study, we focus exclusively on the 10-dimensional instances because none of the tested methods was able to successfully solve the problems even at this lowest dimensionality; therefore, higher-dimensional instances were not considered.

To incorporate the CEC 2014 benchmark into our test problems, we additionally introduce a parameter p to indicate the specific benchmark function being used. Specifically, let

$$p \in \{1, \ldots, 30\}$$

denote the index of the CEC 2014 benchmark problem. For a decision subvector $\mathbf{x}_i \in \mathbb{R}^{10}$, the subproblem objective function is defined as

$$f_{\text{sub}}(10, \mathbf{x}_i, p, i) \;=\; \text{CEC2014}_p(\mathbf{x}_i).$$

The corresponding overall objective function is then defined as

$$fitness(m, 10, p, \mathbf{x}) \;=\; \sum_{i=1}^{m} f_{\text{sub}}(10, \mathbf{x}_i, p, i).$$

doi:10.6342/NTU202600592

4.2 Experiment Setup

All experiments were conducted with 30 independent runs. For each run, the maximum number of fitness evaluations was limited to 400,000. The decision variables were constrained to $[-200, 200]$ in the first test suite and to $[-100, 100]$ in the second test suite. Performance was measured using the absolute error:

$$\text{Error}(\mathbf{x}) = \left| fitness(\mathbf{x}) - fitness(\mathbf{x}^*) \right|,$$

where $\mathbf{x}^*$ denotes the global optimum, and $fitness(\mathbf{x})$ denotes its associated fitness value.

We compared the proposed approaches under several experimental settings, including mSoD + ECGA, smSoD + ECGA, L-SHADE, and RV-GOMEA. The same parameter configuration was used for mSoD + ECGA and smSoD + ECGA (as well as their variants), summarized below:

- Maximum generations: 2000
- Population size: 200
- Crossover probability: 0.975
- Tournament size: 8
- Split rate γ: 0.5
- Split-rate decay factor ε: 0.998

doi:10.6342/NTU202600592

For L-SHADE and RV-GOMEA [21, 2], we followed the parameter settings suggested in their respective original studies.

4.3 Results and Discussions

In this section, we present the experimental results obtained under various experimental scenarios and discuss the corresponding findings. The performance of each method is evaluated based on a comparison of their average rankings across different numbers of subproblems for each problem set. Additionally, we conclude this section by reporting the peak memory usage of both RV-GOMEA and smSoD→mSoD + ECGA.

4.3.1 Performance under Known Linkage Information

In this subsection, we compare L-SHADE, mSoD + ECGA, and smSoD + ECGA. Although L-SHADE is model-based, it does not explicitly exploit dimension-wise information for modeling and thus avoids large memory overhead on large-scale problems. The latter two methods are evaluated under the assumption that accurate linkage information is available. Detailed results are provided in Appendix A.1.

For the decomposable linkage problems, we consider subproblem sizes $k \in \{2,3,4,5\}$ and numbers of subproblems $m \in \{10,20,50,100\}$. As shown in Fig. 4.1, for all considered values of m, smSoD + ECGA consistently achieves better average ranking than mSoD + ECGA across all tested subproblem sizes k. In comparison with L-SHADE, smSoD + ECGA performs significantly worse when m is small (*i.e.*, $m = 10,20,50$) for most values

doi:10.6342/NTU202600592

of k. However, when m increases to 100, smSoD + ECGA achieves better average ranking than L-SHADE for all tested subproblem sizes.

Similarly, Fig. 4.2 presents the experimental results on the extended CEC 2014 benchmark, which exhibits the same trend observed in the decomposable linkage problems. When the number of subproblems m is small, smSoD + ECGA underperforms both mSoD + ECGA and L-SHADE. As m increases, however, the performance of smSoD + ECGA improves steadily and eventually achieves the best average ranking among all compared methods.

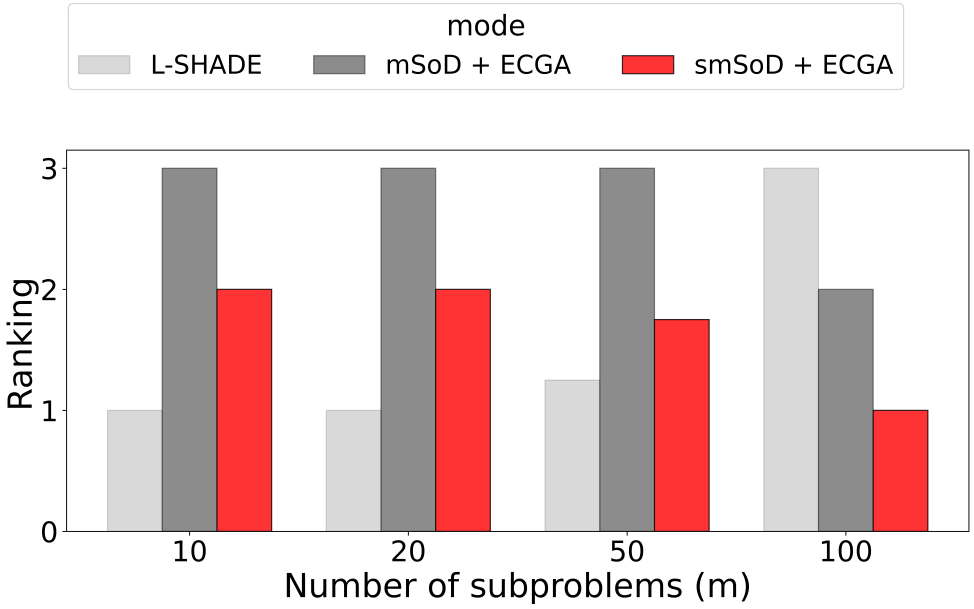


Figure 4.1: Average ranking on decomposable linkage problems with linkage information available across subproblem sizes $k \in \{2,3,4,5\}$ and numbers of subproblems $m \in \{10,20,50,100\}$. Both mSoD and smSoD assume that linkage information is available.

doi:10.6342/NTU202600592

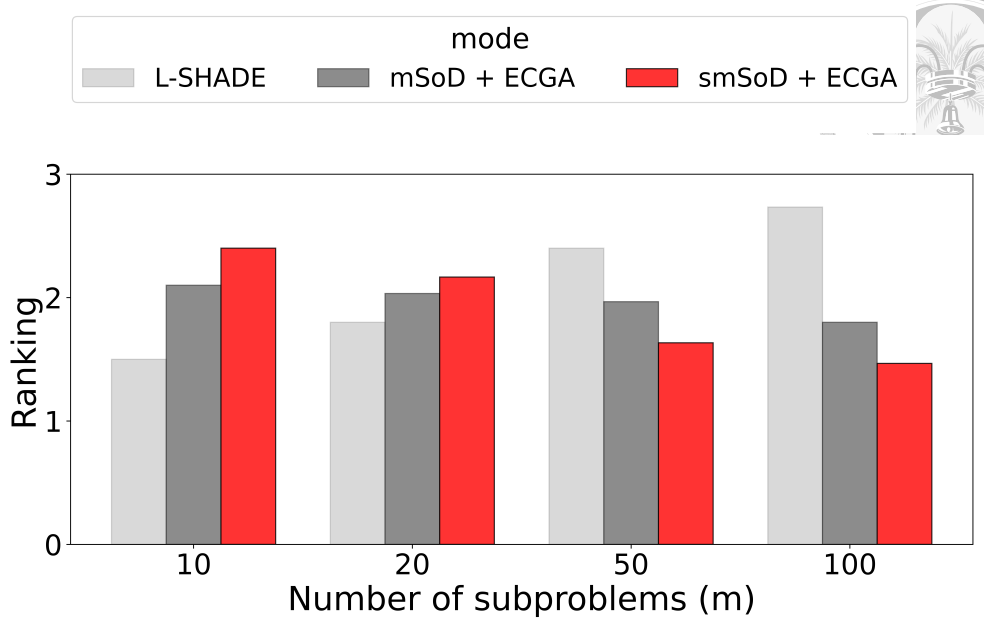


Figure 4.2: Average ranking on the extended CEC 2014 benchmark with linkage information available across numbers of subproblems $m \in \{10, 20, 50, 100\}$. Both mSoD and smSoD assume that linkage information is available.

4.3.2 Performance under Unknown Linkage Information

In this subsection, we first compare four discretization methods—mSoD, smSoD, and their two combinations—when prior linkage information is unavailable. We then select the most competitive method based on the benchmark results and conduct a final comparison against L-SHADE. Detailed results are provided in Appendix A.2.

In this study, we evaluate the following four discretization strategies:

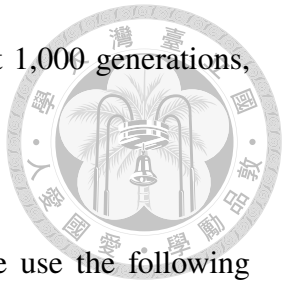
1. **mSoD**: mSoD is applied throughout the entire optimization process.

2. **smSoD**: smSoD is applied throughout the entire optimization process.

3. **mSoD followed by smSoD**: mSoD is applied during the first 1,000 generations, followed by smSoD during the subsequent 1,000 generations.

doi:10.6342/NTU202600592

4. **smSoD followed by mSoD**: smSoD is applied during the first 1,000 generations, followed by mSoD during the subsequent 1,000 generations.



Due to space limitations, in the remainder of this section, we use the following shorthand notation: mSoD followed by smSoD is denoted as mSoD→smSoD, and smSoD followed by mSoD is denoted as smSoD→mSoD.

Under the experimental settings for the decomposable linkage problems, we consider subproblem sizes $k \in \{2,3,4,5\}$ and numbers of subproblems $m \in \{10,20,50,100,200\}$. In the extended CEC 2014 benchmark, we evaluate the cases where $m \in \{10,20,50,100\}$.

As shown in Fig. 4.3 and Fig. 4.4, smSoD→mSoD demonstrates an advantage in terms of average ranking across both benchmarks for most values of m. The only exception occurs in decomposable linkage problems with a small number of subproblems ($m = 10$), where it is outperformed by smSoD alone.

We compare a hybrid approach that combines mSoD and smSoD because, under unknown linkage conditions, we observed a discrepancy: smSoD produces better structure but does not deliver better performance, whereas mSoD yields weaker structure yet performs better. On the CEC 2014 benchmark (30 problems with 100 subproblems), smSoD consistently exhibits stronger structure. We suspect that our split-point selection is biased, leading to imbalanced sampling and premature convergence, which in turn limits performance. Combining this selection strategy with the original random strategy helps restore sampling balance and improves overall results.

Among the four discretization methods, the best-performing approach, smSoD→mSoD,

doi:10.6342/NTU202600592

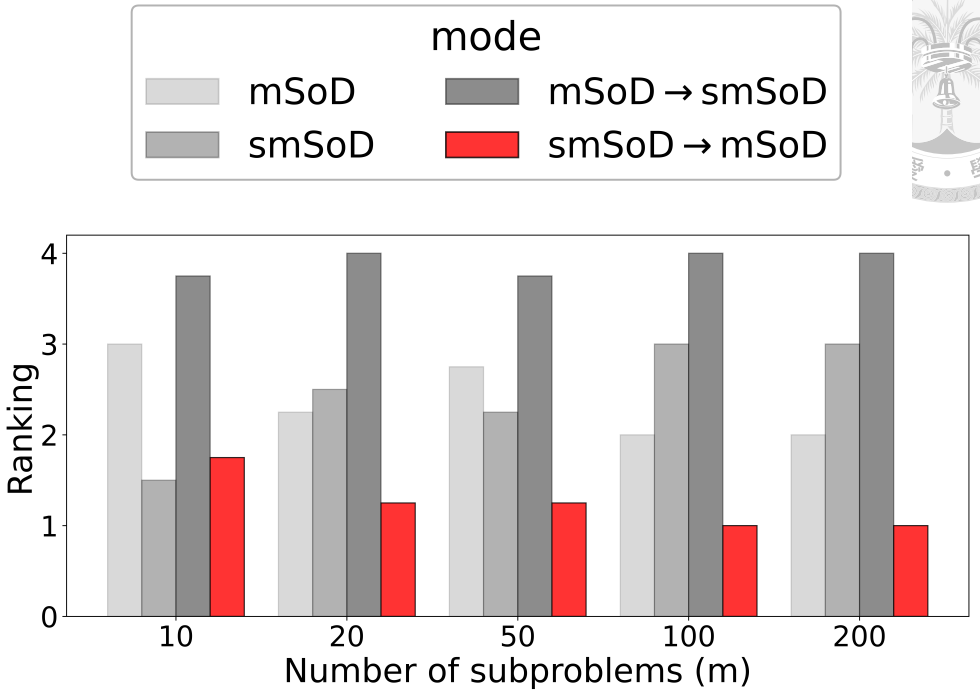


Figure 4.3: Average ranking of four discretization strategies on decomposable linkage problems with unknown linkage information across subproblem sizes $k \in \{2,3,4,5\}$ and numbers of subproblems $m \in \{10,20,50,100,200\}$.

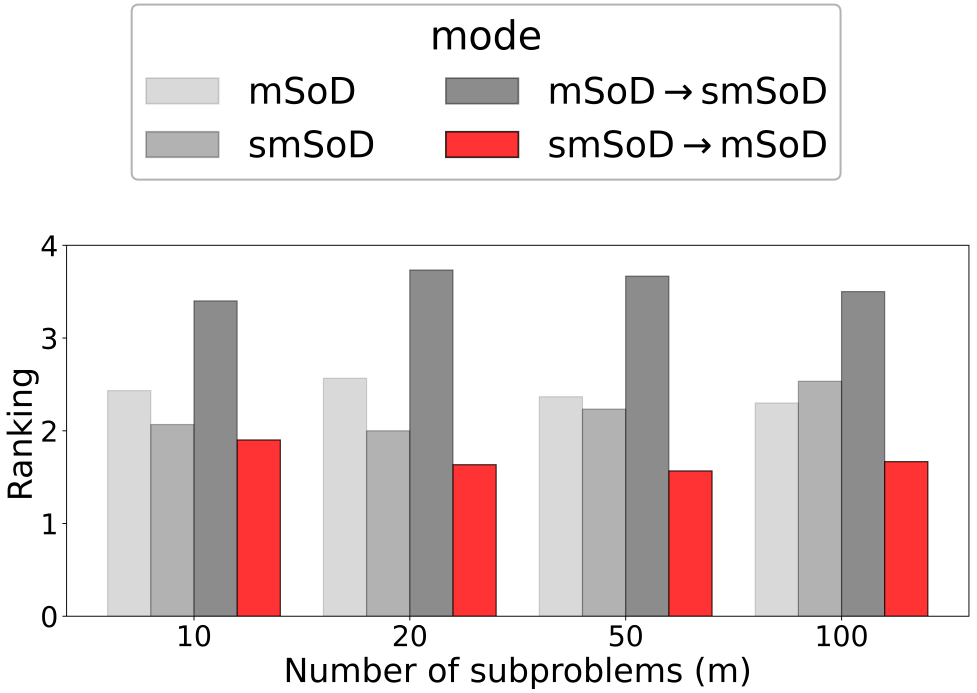


Figure 4.4: Average ranking of four discretization strategies on the extended CEC 2014 benchmark with unknown linkage information across numbers of subproblems $m \in \{10,20,50,100\}$.

doi:10.6342/NTU202600592

is selected for comparison with L-SHADE. The test problem settings used in this experiment are identical to those described above. The experimental results are shown in Fig. 4.5 and Fig. 4.6. It can be observed that smSoD→mSoD + ECGA exhibits a consistent performance trend across the two problem groups. Specifically, when the number of subproblems is small, it does not demonstrate a clear advantage. However, as the number of subproblems increases, smSoD→mSoD + ECGA achieves better average performance than L-SHADE on most test problems.

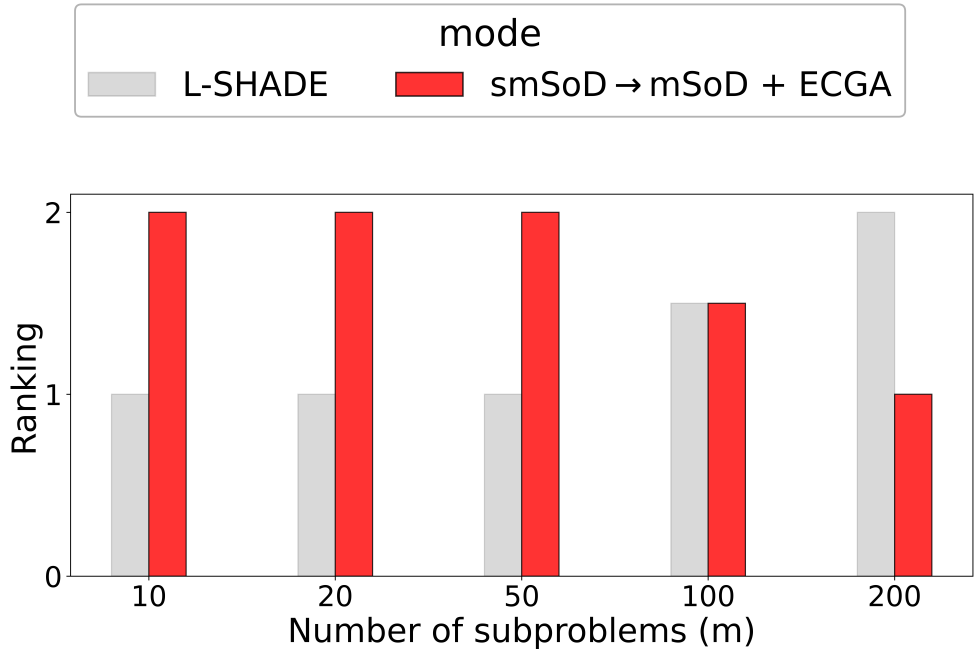


Figure 4.5: Average ranking comparison between L-SHADE and smSoD→mSoD + ECGA on decomposable linkage problems with unknown linkage information across subproblem sizes $k \in \{2,3,4,5\}$ and numbers of subproblems $m \in \{10,20,50,100,200\}$.

4.3.3 Performance under Memory Constraints

In this subsection, we compare smSoD→mSoD + ECGA with another model-building genetic algorithm, RV-GOMEA, with a particular focus on their behavior under constrained

doi:10.6342/NTU202600592

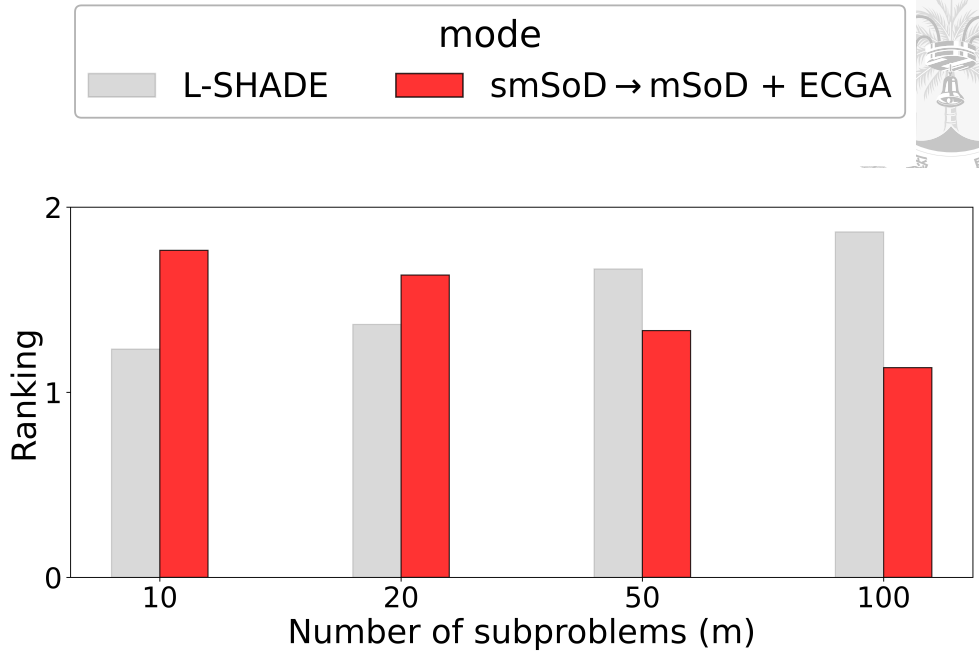


Figure 4.6: Average ranking comparison between L-SHADE and smSoD→mSoD + ECGA on the extended CEC 2014 benchmark with unknown linkage information across numbers of subproblems $m \in \{10, 20, 50, 100\}$.

memory budgets. Specifically, we evaluate both methods under five memory configurations: 64, 128, 256, 512, and 1024 MB. Memory limits are enforced via Docker; whenever memory consumption exceeds the prescribed budget, the operating system terminates the process via an out-of-memory (OOM) kill. All experiments are conducted on the extended CEC 2014 benchmark, with the number of subproblems fixed at $m = 100$.

The results are shown in Fig. 4.7. Across all five memory budgets, smSoD→mSoD + ECGA consistently achieves a better average ranking than RV-GOMEA. As more memory becomes available, however, RV-GOMEA gradually becomes competitive and improves its average ranking. A plausible explanation is that the underlying optimization model in smSoD→mSoD + ECGA is relatively compact, whereas RV-GOMEA employs a more expressive probabilistic model. Consequently, with increasing memory budgets,

doi:10.6342/NTU202600592

RV-GOMEA is able to better exploit its modeling capacity and thus achieve improved performance. Detailed results are provided in Table 4.1, where performance is reported in terms of the mean.

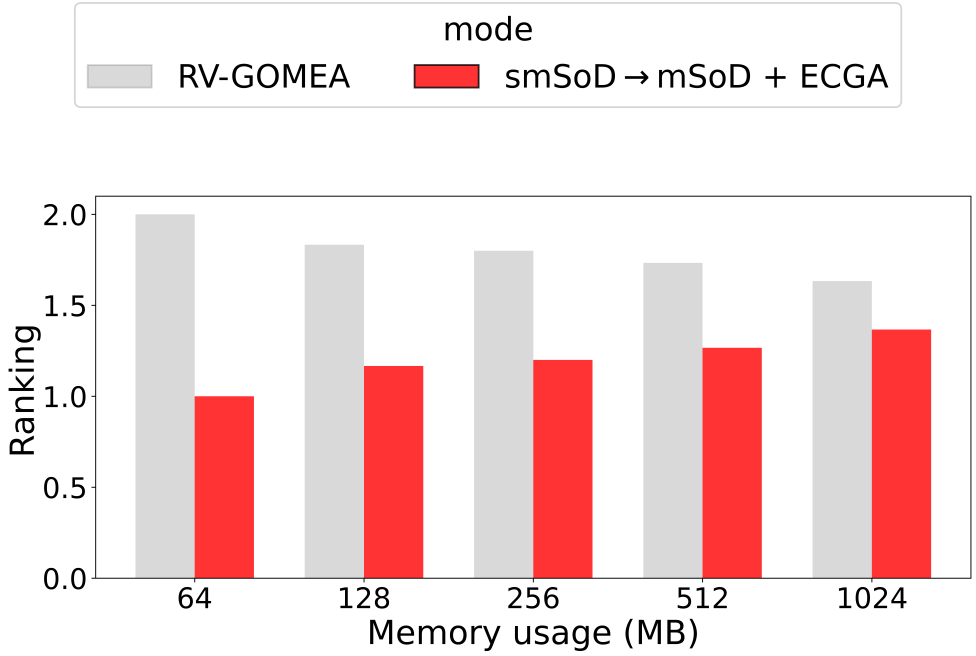


Figure 4.7: Average ranking comparison between RV-GOMEA and smSoD→mSoD + ECGA on the extended CEC 2014 benchmark under constrained memory budgets with the number of subproblems $m = 100$.

doi:10.6342/NTU202600592

Table 4.1: Mean error comparison between RV-GOMEA and smSoD→mSoD + ECGA on the extended CEC 2014 benchmark under constrained memory budgets with the number of subproblems $m = 100$. In this table, smSoD→mSoD + ECGA is denoted as smSoD→mSoD.

	64 MB		128 MB		256 MB	
Function	RV-GOMEA	smSoD→mSoD	RV-GOMEA	smSoD→mSoD	RV-GOMEA	smSoD→mSoD
1	5.02×10^{9}	$\mathbf{2.87 \times 10^{9}}$	3.11×10^{9}	$\mathbf{2.87 \times 10^{9}}$	3.24×10^{9}	$\mathbf{2.87 \times 10^{9}}$
2	7.45×10^{11}	$\mathbf{1.91 \times 10^{9}}$	$\mathbf{2.27 \times 10^{8}}$	1.91×10^{9}	$\mathbf{4.47 \times 10^{8}}$	1.91×10^{9}
3	1.31×10^{8}	$\mathbf{7.96 \times 10^{5}}$	1.70×10^{8}	$\mathbf{7.96 \times 10^{5}}$	7.76×10^{7}	$\mathbf{7.96 \times 10^{5}}$
4	2.26×10^{4}	$\mathbf{5.96 \times 10^{3}}$	$\mathbf{4.49 \times 10^{3}}$	5.96×10^{3}	$\mathbf{4.19 \times 10^{3}}$	5.96×10^{3}
5	2.11×10^{3}	$\mathbf{2.00 \times 10^{3}}$	2.11×10^{3}	$\mathbf{2.00 \times 10^{3}}$	2.11×10^{3}	$\mathbf{2.00 \times 10^{3}}$
6	5.69×10^{2}	$\mathbf{3.89 \times 10^{2}}$	5.35×10^{2}	$\mathbf{3.89 \times 10^{2}}$	5.51×10^{2}	$\mathbf{3.89 \times 10^{2}}$
7	1.35×10^{3}	$\mathbf{1.18 \times 10^{2}}$	$\mathbf{1.12 \times 10^{2}}$	1.18×10^{2}	$\mathbf{1.04 \times 10^{2}}$	1.18×10^{2}
8	2.26×10^{3}	$\mathbf{1.08 \times 10^{3}}$	2.15×10^{3}	$\mathbf{1.08 \times 10^{3}}$	2.05×10^{3}	$\mathbf{1.08 \times 10^{3}}$
9	4.13×10^{3}	$\mathbf{1.44 \times 10^{3}}$	7.63×10^{3}	$\mathbf{1.44 \times 10^{3}}$	6.04×10^{3}	$\mathbf{1.44 \times 10^{3}}$
10	4.32×10^{4}	$\mathbf{2.90 \times 10^{4}}$	4.11×10^{4}	$\mathbf{2.90 \times 10^{4}}$	4.06×10^{4}	$\mathbf{2.90 \times 10^{4}}$
11	2.69×10^{5}	$\mathbf{9.12 \times 10^{4}}$	2.60×10^{5}	$\mathbf{9.12 \times 10^{4}}$	2.66×10^{5}	$\mathbf{9.12 \times 10^{4}}$
12	6.10×10^{2}	$\mathbf{5.69 \times 10^{1}}$	6.25×10^{2}	$\mathbf{5.69 \times 10^{1}}$	5.55×10^{2}	$\mathbf{5.69 \times 10^{1}}$
13	1.27×10^{2}	$\mathbf{3.51 \times 10^{1}}$	1.14×10^{2}	$\mathbf{3.51 \times 10^{1}}$	1.17×10^{2}	$\mathbf{3.51 \times 10^{1}}$
14	3.16×10^{2}	$\mathbf{5.33 \times 10^{1}}$	1.81×10^{2}	$\mathbf{5.33 \times 10^{1}}$	1.79×10^{2}	$\mathbf{5.33 \times 10^{1}}$
15	7.67×10^{5}	$\mathbf{2.37 \times 10^{4}}$	$\mathbf{5.09 \times 10^{3}}$	2.37×10^{4}	$\mathbf{4.16 \times 10^{3}}$	2.37×10^{4}
16	4.61×10^{2}	$\mathbf{4.30 \times 10^{2}}$	4.57×10^{2}	$\mathbf{4.30 \times 10^{2}}$	4.53×10^{2}	$\mathbf{4.30 \times 10^{2}}$
17	2.73×10^{9}	$\mathbf{8.81 \times 10^{8}}$	2.33×10^{9}	$\mathbf{8.81 \times 10^{8}}$	1.03×10^{9}	$\mathbf{8.81 \times 10^{8}}$
18	2.49×10^{8}	$\mathbf{8.97 \times 10^{5}}$	7.63×10^{6}	$\mathbf{8.97 \times 10^{5}}$	1.49×10^{6}	$\mathbf{8.97 \times 10^{5}}$
19	1.19×10^{3}	$\mathbf{3.06 \times 10^{2}}$	4.48×10^{2}	$\mathbf{3.06 \times 10^{2}}$	3.99×10^{2}	$\mathbf{3.06 \times 10^{2}}$
20	1.38×10^{9}	$\mathbf{7.61 \times 10^{5}}$	5.92×10^{7}	$\mathbf{7.61 \times 10^{5}}$	4.60×10^{7}	$\mathbf{7.61 \times 10^{5}}$
21	4.97×10^{8}	$\mathbf{2.62 \times 10^{8}}$	4.78×10^{8}	$\mathbf{2.62 \times 10^{8}}$	4.93×10^{8}	$\mathbf{2.62 \times 10^{8}}$
22	4.10×10^{4}	$\mathbf{1.77 \times 10^{4}}$	3.64×10^{4}	$\mathbf{1.77 \times 10^{4}}$	3.33×10^{4}	$\mathbf{1.77 \times 10^{4}}$
23	3.97×10^{4}	$\mathbf{3.35 \times 10^{4}}$	$\mathbf{3.29 \times 10^{4}}$	3.35×10^{4}	$\mathbf{3.28 \times 10^{4}}$	3.35×10^{4}
24	1.86×10^{4}	$\mathbf{1.71 \times 10^{4}}$	1.84×10^{4}	$\mathbf{1.71 \times 10^{4}}$	1.85×10^{4}	$\mathbf{1.71 \times 10^{4}}$
25	2.03×10^{4}	$\mathbf{1.97 \times 10^{4}}$	1.99×10^{4}	$\mathbf{1.97 \times 10^{4}}$	2.01×10^{4}	$\mathbf{1.97 \times 10^{4}}$
26	1.31×10^{4}	$\mathbf{1.19 \times 10^{4}}$	1.31×10^{4}	$\mathbf{1.19 \times 10^{4}}$	1.32×10^{4}	$\mathbf{1.19 \times 10^{4}}$
27	4.51×10^{4}	$\mathbf{4.27 \times 10^{4}}$	4.56×10^{4}	$\mathbf{4.27 \times 10^{4}}$	4.45×10^{4}	$\mathbf{4.27 \times 10^{4}}$
28	1.23×10^{5}	$\mathbf{5.46 \times 10^{4}}$	6.50×10^{4}	$\mathbf{5.46 \times 10^{4}}$	6.51×10^{4}	$\mathbf{5.46 \times 10^{4}}$
29	8.39×10^{7}	$\mathbf{4.71 \times 10^{7}}$	5.80×10^{7}	$\mathbf{4.71 \times 10^{7}}$	5.53×10^{7}	$\mathbf{4.71 \times 10^{7}}$
30	3.23×10^{6}	$\mathbf{9.13 \times 10^{5}}$	1.08×10^{6}	$\mathbf{9.13 \times 10^{5}}$	$\mathbf{5.38 \times 10^{5}}$	9.13×10^{5}

doi:10.6342/NTU202600592

Table 4.1: (continued)

	512 MB		1024 MB	
Function	RV-GOMEA	smSoD→mSoD	RV-GOMEA	smSoD→mSoD
1	$\mathbf{2.34 \times 10^9}$	2.87×10^9	$\mathbf{1.19 \times 10^9}$	2.87×10^9
2	$\mathbf{8.23 \times 10^7}$	1.91×10^9	$\mathbf{3.41 \times 10^5}$	1.91×10^9
3	1.22×10^6	$\mathbf{7.96 \times 10^5}$	$\mathbf{7.42 \times 10^5}$	7.96×10^5
4	$\mathbf{3.64 \times 10^3}$	5.96×10^3	$\mathbf{2.91 \times 10^3}$	5.96×10^3
5	2.11×10^3	$\mathbf{2.00 \times 10^3}$	2.11×10^3	$\mathbf{2.00 \times 10^3}$
6	5.48×10^2	$\mathbf{3.89 \times 10^2}$	5.32×10^2	$\mathbf{3.89 \times 10^2}$
7	$\mathbf{1.06 \times 10^2}$	1.18×10^2	$\mathbf{7.90 \times 10^1}$	1.18×10^2
8	1.90×10^3	$\mathbf{1.08 \times 10^3}$	1.95×10^3	$\mathbf{1.08 \times 10^3}$
9	3.60×10^3	$\mathbf{1.44 \times 10^3}$	2.90×10^3	$\mathbf{1.44 \times 10^3}$
10	3.71×10^4	$\mathbf{2.90 \times 10^4}$	3.43×10^4	$\mathbf{2.90 \times 10^4}$
11	2.64×10^5	$\mathbf{9.12 \times 10^4}$	1.08×10^5	$\mathbf{9.12 \times 10^4}$
12	5.61×10^2	$\mathbf{5.69 \times 10^1}$	5.27×10^2	$\mathbf{5.69 \times 10^1}$
13	1.03×10^2	$\mathbf{3.51 \times 10^1}$	8.66×10^1	$\mathbf{3.51 \times 10^1}$
14	1.59×10^2	$\mathbf{5.33 \times 10^1}$	1.23×10^2	$\mathbf{5.33 \times 10^1}$
15	$\mathbf{3.54 \times 10^3}$	2.37×10^4	$\mathbf{1.76 \times 10^3}$	2.37×10^4
16	4.51×10^2	$\mathbf{4.30 \times 10^2}$	4.44×10^2	$\mathbf{4.30 \times 10^2}$
17	$\mathbf{5.78 \times 10^8}$	8.81×10^8	$\mathbf{3.97 \times 10^8}$	8.81×10^8
18	1.01×10^6	$\mathbf{8.97 \times 10^5}$	1.04×10^6	$\mathbf{8.97 \times 10^5}$
19	3.59×10^2	$\mathbf{3.06 \times 10^2}$	3.23×10^2	$\mathbf{3.06 \times 10^2}$
20	9.07×10^5	$\mathbf{7.61 \times 10^5}$	$\mathbf{7.18 \times 10^5}$	7.61×10^5
21	4.19×10^8	$\mathbf{2.62 \times 10^8}$	3.50×10^8	$\mathbf{2.62 \times 10^8}$
22	2.19×10^4	$\mathbf{1.77 \times 10^4}$	1.96×10^4	$\mathbf{1.77 \times 10^4}$
23	$\mathbf{3.28 \times 10^4}$	3.35×10^4	$\mathbf{3.23 \times 10^4}$	3.35×10^4
24	1.80×10^4	$\mathbf{1.71 \times 10^4}$	1.73×10^4	$\mathbf{1.71 \times 10^4}$
25	1.99×10^4	$\mathbf{1.97 \times 10^4}$	1.98×10^4	$\mathbf{1.97 \times 10^4}$
26	1.29×10^4	$\mathbf{1.19 \times 10^4}$	1.23×10^4	$\mathbf{1.19 \times 10^4}$
27	4.45×10^4	$\mathbf{4.27 \times 10^4}$	4.44×10^4	$\mathbf{4.27 \times 10^4}$
28	6.13×10^4	$\mathbf{5.46 \times 10^4}$	6.02×10^4	$\mathbf{5.46 \times 10^4}$
29	4.88×10^7	$\mathbf{4.71 \times 10^7}$	$\mathbf{3.30 \times 10^7}$	4.71×10^7
30	$\mathbf{7.63 \times 10^5}$	9.13×10^5	$\mathbf{1.75 \times 10^5}$	9.13×10^5

doi:10.6342/NTU202600592

4.3.4 Peak Memory Usage at Different Dimensionalities

This subsection reports the peak memory usage of RV-GOMEA and smSoD→mSoD + ECGA. Figure 4.8 presents the peak memory consumption of the two methods under different dimensionalities (500, 1000, 2000, and 4000). All experiments were conducted on a machine equipped with 64 GB of RAM. At a dimensionality of 4000, RV-GOMEA terminates due to an out-of-memory (OOM) error, indicating the rapid growth of the memory footprint of its internal data structures. In contrast, smSoD→mSoD + ECGA completes successfully even at a dimensionality of 4000 under the same 64 GB memory budget, suggesting that our method is more scalable and applicable to high-dimensional problems.

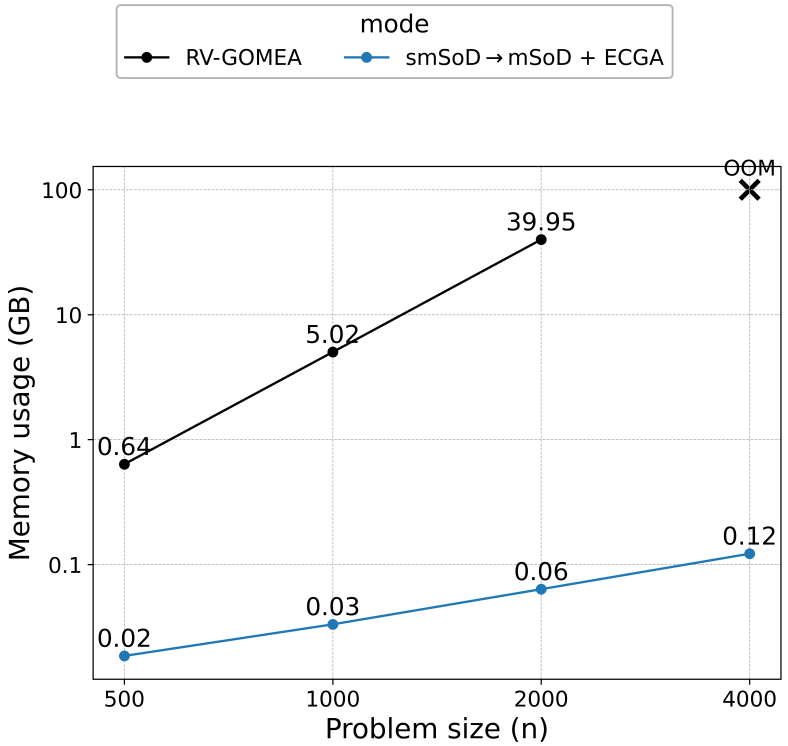


Figure 4.8: Peak memory usage of RV-GOMEA and smSoD→mSoD + ECGA when optimizing problems of different dimensionalities.

doi:10.6342/NTU202600592

4.4 Computational Overhead

This section examines the computational overhead of mSoD and smSoD when linkage information is available. We quantify the overhead by counting the total number of executed operations using likwid-perfctr [22].

On decomposable linkage problems, smSoD incurs a higher number of operations than mSoD. When the number of subproblems is $m = 10$, smSoD executes approximately 343% more operations than mSoD. As m increases, the additional overhead decreases, dropping to about 117% more operations at $m = 100$.

A similar trend is also evident on the extended CEC 2014 benchmark. At $m = 10$, smSoD performs roughly 141% more operations than mSoD, and this gap narrows to around 91% more operations when $m = 50$.

doi:10.6342/NTU202600592

Chapter 5

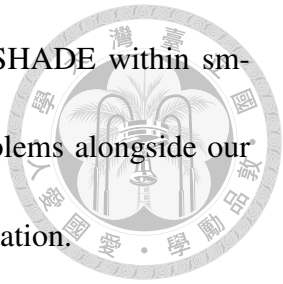
Conclusion

In this work, we introduced smSoD, a variant of the original mSoD algorithm that enhances the split-point selection strategy. Although mSoD traditionally requires prior linkage information to achieve strong performance, our experimental results showed that smSoD could be effectively integrated with ECGA to learn linkage information when it was unavailable. This integration enabled the algorithm to infer linkage structures automatically, thereby eliminating the need for predefined linkage information.

Across experiments with known linkage information, unknown linkage information, and memory-constrained settings, we observed the following trends. In the first two scenarios, when the number of subproblems was sufficiently large, our approach achieved a better average ranking than the original mSoD and demonstrated the same advantage over L-SHADE. In the memory-constrained scenario, RV-GOMEA exhibited substantial overhead due to the large memory footprint of its modeling data structures; consequently, in high-dimensional settings with limited memory budgets, our method attained a superior average ranking.

doi:10.6342/NTU202600592

As for future work, we plan to study the incorporation of L-SHADE within sm-SoD. Exploiting L-SHADE's effectiveness on low-dimensional problems alongside our framework could produce notable gains for high-dimensional optimization.



doi:10.6342/NTU202600592

References

[1] Peter A. N. Bosman and Dirk Thierens. Linkage neighbors, optimal mixing and forced improvements in genetic algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 585–592, 2012.

[2] Anton Bouter, Tanja Alderliesten, Cees Witteveen, and Peter A. N. Bosman. Exploiting linkage information in real-valued optimization with the real-valued gene-pool optimal mixing evolutionary algorithm. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 705–712, 2017.

[3] Chao-Hong Chen, Wei-Nan Liu, and Ying-ping Chen. Adaptive discretization for probabilistic model building genetic algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1103–1110, 2006.

[4] Amit Choudhury, Subhasis Ray, and Pradipta Sarkar. Approximating the cumulative distribution function of the normal distribution. *Journal of Statistical Research*, 41:59–67, 2007.

doi:10.6342/NTU202600592

[5] Kenneth Alan De Jong. *An Analysis of the Behavior of a Class of Genetic Adaptive Systems*. PhD thesis, University of Michigan, 1975.

[6] Kalyanmoy Deb and David E. Goldberg. Sufficient conditions for deceptive and easy binary functions. *Annals of Mathematics and Artificial Intelligence*, 10:385–408, 1994.

[7] Agoston E Eiben and James E Smith. *Introduction to evolutionary computing*. Springer, 2015.

[8] Iztok Fister, Suash Deb, Dusan Fister, and Iztok Fister Jr. How does selecting a benchmark function suite influence the estimation of an algorithm's quality? In *Proceedings of the International Conference on Soft Computing & Machine Intelligence*, 2019.

[9] Georges R. Harik, Fernando G. Lobo, and Kumara Sastry. Linkage learning via probabilistic modeling in the extended compact genetic algorithm (ECGA). In *Scalable Optimization via Probabilistic Modeling*, pages 39–61. Springer, 2006.

[10] John H. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. MIT Press, 1992.

[11] Ping-Chu Hung and Ying-ping Chen. iECGA: Integer extended compact genetic algorithm. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1415–1416, 2006.

doi:10.6342/NTU202600592

[12] Pedro Larrañaga and Jose A Lozano. *Estimation of distribution algorithms: A new tool for evolutionary computation*, volume 2. Springer, 2001.

[13] Jing J Liang, Bo Y Qu, Ponnuthurai N Suganthan, et al. Problem definitions and evaluation criteria for the CEC 2014 special session and competition on single objective real-parameter numerical optimization. *Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University, Singapore*, 635(2):2014, 2013.

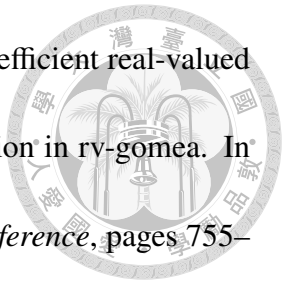
[14] Jiun-Jiue Liou and Ying-ping Chen. Adaptive discretization on multidimensional continuous search spaces. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 977–984, 2008.

[15] Martin Pelikan, David E. Goldberg, and Fernando G. Lobo. A survey of optimization by building and using probabilistic models. *Computational Optimization and Applications*, 21:5–20, 2002.

[16] Adam P Piotrowski, Jaroslaw J Napiorkowski, and Agnieszka E Piotrowska. Choice of benchmark optimization problems does matter. *Swarm and Evolutionary Computation*, 83:101378, 2023.

[17] James A. Preiss, Wolfgang Hönig, Nora Ayanian, and Gaurav S. Sukhatme. Downwash-aware trajectory planning for large quadrotor teams. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 250–257. IEEE/RSJ, 2017.

doi:10.6342/NTU202600592



[18] Renzo Scholman, Tanja Alderliesten, and Peter Bosman. More efficient real-valued gray-box optimization through incremental distribution estimation in rv-gomea. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 755–763, 2025.

[19] Rainer Storn and Kenneth Price. Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11:341–359, 1997.

[20] Ryoji Tanabe and Alex Fukunaga. Success-history based parameter adaptation for differential evolution. In *IEEE Congress on Evolutionary Computation*, pages 71–78. IEEE, 2013.

[21] Ryoji Tanabe and Alex S. Fukunaga. Improving the search performance of SHADE using linear population size reduction. In *IEEE Congress on Evolutionary Computation*, pages 1658–1665. IEEE, 2014.

[22] Jan Treibig, Georg Hager, and Gerhard Wellein. LIKWID: Lightweight performance tools. In *Competence in High Performance Computing*, pages 165–175. Springer, 2011.

[23] Daniel A. Winkler, Robert Wang, Francois Blanchette, Miguel Carreira-Perpinán, and Alberto E. Cerpa. MAGIC: Model-based actuation for ground irrigation control. In *ACM/IEEE International Conference on Information Processing in Sensor Networks*, pages 1–12. IEEE, 2016.

doi:10.6342/NTU202600592

Appendix

In this appendix, we present detailed experimental results for two of the three scenarios described in Section 4.3: known linkage information and unknown linkage information. All results are reported as mean values.

A.1 Performance on Problem under Known Linkage Information

This section reports the detailed experimental results of L-SHADE, mSoD + ECGA, and smSoD + ECGA under the assumption that linkage information is available. The results are provided for two benchmark suites: the decomposable linkage problems (see Table A.1) and the extended CEC 2014 benchmark (see Table A.2). In all tables, we report performance in terms of the mean.

doi:10.6342/NTU202600592

Table A.1: Mean error on decomposable linkage problems with linkage information available across subproblem sizes $k \in \{2,3,4,5\}$ and numbers of subproblems $m \in \{10,20,50,100\}$. Both mSoD and smSoD assume that linkage information is available.

(a) $k = 2$ and $k = 3$

	$k=2$			$k=3$		
m	L-SHADE	mSoD + ECGA	smSoD + ECGA	L-SHADE	mSoD + ECGA	smSoD + ECGA
10	$\mathbf{0.00 \times 10^{0}}$	1.36×10^{-21}	4.66×10^{-25}	$\mathbf{0.00 \times 10^{0}}$	1.04×10^{-9}	5.60×10^{-18}
20	$\mathbf{0.00 \times 10^{0}}$	1.01×10^{-13}	1.68×10^{-23}	$\mathbf{2.81 \times 10^{-25}}$	2.38×10^{-4}	1.04×10^{-8}
50	$\mathbf{1.28 \times 10^{-12}}$	2.62×10^{-4}	2.82×10^{-10}	1.17×10^{-1}	7.63×10^{-1}	$\mathbf{5.23 \times 10^{-3}}$
100	1.14×10^{0}	5.66×10^{-3}	$\mathbf{3.52 \times 10^{-6}}$	6.11×10^{3}	1.61×10^{3}	$\mathbf{2.37 \times 10^{2}}$

(b) $k = 4$ and $k = 5$

	$k=4$			$k=5$		
m	L-SHADE	mSoD + ECGA	smSoD + ECGA	L-SHADE	mSoD + ECGA	smSoD + ECGA
10	$\mathbf{0.00 \times 10^{0}}$	1.43×10^{-4}	3.71×10^{-7}	$\mathbf{2.45 \times 10^{-24}}$	1.09×10^{-1}	3.31×10^{-2}
20	$\mathbf{1.08 \times 10^{-9}}$	2.01×10^{-1}	6.79×10^{-3}	$\mathbf{2.15 \times 10^{-3}}$	2.63×10^{2}	6.12×10^{1}
50	$\mathbf{3.25 \times 10^{2}}$	2.28×10^{3}	4.27×10^{2}	$\mathbf{8.06 \times 10^{3}}$	2.64×10^{4}	9.89×10^{3}
100	1.17×10^{5}	5.21×10^{4}	$\mathbf{1.91 \times 10^{4}}$	2.94×10^{5}	1.73×10^{5}	$\mathbf{1.06 \times 10^{5}}$

A.2 Performance on Problem under Unknown Linkage Information

This section reports detailed experimental results for mSoD, smSoD, mSoD followed by smSoD, and smSoD followed by mSoD under the assumption that linkage information is unavailable. Results are provided for two benchmark suites: the decomposable linkage problems (see Table A.3) and the extended CEC 2014 benchmark (see Table A.4). We then select the best-performing discretization strategy—smSoD followed by mSoD—and compare it with L-SHADE; the results are summarized in Table A.5 and Table A.6. In all tables, we report performance in terms of the mean. Due to space limitations, in the remainder of this section, we use the following shorthand notation: mSoD followed

doi:10.6342/NTU202600592

Table A.2: Mean error on the extended CEC 2014 benchmark with linkage information available across numbers of subproblems $m \in \{10, 20, 50, 100\}$. Both mSoD and smSoD assume that linkage information is available.

	$m = 10$			$m = 20$		
Function	L-SHADE	mSoD + ECGA	smSoD + ECGA	L-SHADE	mSoD + ECGA	smSoD + ECGA
1	$\mathbf{2.37 \times 10^{5}}$	1.91×10^{7}	8.19×10^{6}	$\mathbf{1.63 \times 10^{7}}$	1.98×10^{8}	1.33×10^{8}
2	$\mathbf{7.19 \times 10^{2}}$	2.11×10^{5}	4.24×10^{4}	$\mathbf{7.06 \times 10^{3}}$	1.97×10^{6}	1.40×10^{5}
3	$\mathbf{1.15 \times 10^{2}}$	4.79×10^{5}	3.74×10^{5}	$\mathbf{1.02 \times 10^{5}}$	1.29×10^{6}	1.13×10^{6}
4	3.43×10^{2}	2.60×10^{2}	$\mathbf{2.59 \times 10^{2}}$	6.62×10^{2}	6.49×10^{2}	$\mathbf{5.96 \times 10^{2}}$
5	2.02×10^{2}	2.01×10^{2}	$\mathbf{2.00 \times 10^{2}}$	4.15×10^{2}	4.06×10^{2}	$\mathbf{4.00 \times 10^{2}}$
6	$\mathbf{1.32 \times 10^{1}}$	5.51×10^{1}	6.68×10^{1}	$\mathbf{5.64 \times 10^{1}}$	1.25×10^{2}	1.39×10^{2}
7	$\mathbf{1.65 \times 10^{-1}}$	3.91×10^{0}	3.16×10^{0}	$\mathbf{4.14 \times 10^{0}}$	1.41×10^{1}	6.83×10^{0}
8	$\mathbf{5.60 \times 10^{1}}$	9.90×10^{1}	1.38×10^{2}	7.75×10^{2}	$\mathbf{2.56 \times 10^{2}}$	4.21×10^{2}
9	$\mathbf{1.31 \times 10^{2}}$	3.76×10^{2}	4.57×10^{2}	$\mathbf{2.90 \times 10^{2}}$	8.09×10^{2}	1.01×10^{3}
10	2.68×10^{3}	$\mathbf{1.54 \times 10^{3}}$	5.83×10^{3}	3.24×10^{4}	$\mathbf{4.15 \times 10^{3}}$	1.85×10^{4}
11	1.22×10^{4}	$\mathbf{1.02 \times 10^{4}}$	1.12×10^{4}	4.13×10^{4}	$\mathbf{2.20 \times 10^{4}}$	2.36×10^{4}
12	1.33×10^{1}	$\mathbf{4.94 \times 10^{0}}$	5.69×10^{0}	6.15×10^{1}	1.39×10^{1}	$\mathbf{1.28 \times 10^{1}}$
13	$\mathbf{4.22 \times 10^{-1}}$	6.26×10^{0}	6.09×10^{0}	$\mathbf{2.70 \times 10^{0}}$	1.33×10^{1}	1.24×10^{1}
14	$\mathbf{4.90 \times 10^{0}}$	5.67×10^{0}	5.55×10^{0}	$\mathbf{9.30 \times 10^{0}}$	1.21×10^{1}	1.18×10^{1}
15	$\mathbf{2.16 \times 10^{1}}$	3.69×10^{1}	4.94×10^{1}	$\mathbf{9.67 \times 10^{1}}$	1.12×10^{2}	1.05×10^{2}
16	$\mathbf{3.64 \times 10^{1}}$	3.66×10^{1}	3.80×10^{1}	8.61×10^{1}	$\mathbf{7.64 \times 10^{1}}$	7.89×10^{1}
17	$\mathbf{5.97 \times 10^{3}}$	3.90×10^{6}	5.65×10^{6}	$\mathbf{2.83 \times 10^{5}}$	1.52×10^{7}	1.55×10^{7}
18	$\mathbf{5.50 \times 10^{2}}$	1.04×10^{5}	9.55×10^{4}	$\mathbf{5.15 \times 10^{4}}$	2.08×10^{5}	1.96×10^{5}
19	4.10×10^{1}	$\mathbf{1.82 \times 10^{1}}$	2.83×10^{1}	8.33×10^{1}	$\mathbf{5.26 \times 10^{1}}$	9.63×10^{1}
20	$\mathbf{3.53 \times 10^{2}}$	7.89×10^{4}	6.50×10^{4}	$\mathbf{2.27 \times 10^{4}}$	1.81×10^{5}	1.40×10^{5}
21	$\mathbf{2.27 \times 10^{3}}$	9.29×10^{4}	5.08×10^{5}	$\mathbf{6.60 \times 10^{4}}$	1.80×10^{6}	9.04×10^{6}
22	1.26×10^{3}	$\mathbf{7.57 \times 10^{2}}$	1.46×10^{3}	7.59×10^{3}	$\mathbf{2.31 \times 10^{3}}$	4.46×10^{3}
23	$\mathbf{3.29 \times 10^{3}}$	3.30×10^{3}	$\mathbf{3.29 \times 10^{3}}$	$\mathbf{6.59 \times 10^{3}}$	6.68×10^{3}	6.62×10^{3}
24	$\mathbf{1.34 \times 10^{3}}$	1.55×10^{3}	1.68×10^{3}	3.90×10^{3}	$\mathbf{3.26 \times 10^{3}}$	3.47×10^{3}
25	2.00×10^{3}	$\mathbf{1.96 \times 10^{3}}$	2.00×10^{3}	3.99×10^{3}	$\mathbf{3.96 \times 10^{3}}$	3.98×10^{3}
26	$\mathbf{1.00 \times 10^{3}}$	1.01×10^{3}	1.01×10^{3}	$\mathbf{2.00 \times 10^{3}}$	2.01×10^{3}	2.02×10^{3}
27	3.28×10^{3}	$\mathbf{2.22 \times 10^{3}}$	3.53×10^{3}	7.16×10^{3}	$\mathbf{6.45 \times 10^{3}}$	6.84×10^{3}
28	$\mathbf{3.94 \times 10^{3}}$	5.62×10^{3}	6.51×10^{3}	$\mathbf{9.00 \times 10^{3}}$	1.22×10^{4}	1.38×10^{4}
29	$\mathbf{5.69 \times 10^{5}}$	1.71×10^{6}	3.86×10^{6}	9.88×10^{6}	$\mathbf{7.28 \times 10^{6}}$	1.08×10^{7}
30	$\mathbf{9.85 \times 10^{3}}$	2.00×10^{4}	2.04×10^{4}	$\mathbf{3.33 \times 10^{4}}$	4.88×10^{4}	4.50×10^{4}

doi:10.6342/NTU202600592

Table A.2: (continued)

	$m = 50$			$m = 100$		
Function	L-SHADE	mSoD + ECGA	smSoD + ECGA	L-SHADE	mSoD + ECGA	smSoD + ECGA
1	1.11×10^{9}	1.31×10^{9}	$\mathbf{1.10 \times 10^{9}}$	4.51×10^{9}	4.09×10^{9}	$\mathbf{3.53 \times 10^{9}}$
2	1.59×10^{9}	2.28×10^{8}	$\mathbf{5.90 \times 10^{6}}$	2.03×10^{11}	1.89×10^{10}	$\mathbf{5.17 \times 10^{9}}$
3	$\mathbf{8.91 \times 10^{5}}$	3.44×10^{6}	3.32×10^{6}	$\mathbf{1.87 \times 10^{6}}$	7.26×10^{6}	6.71×10^{6}
4	2.60×10^{3}	2.40×10^{3}	$\mathbf{2.12 \times 10^{3}}$	3.10×10^{4}	8.87×10^{3}	$\mathbf{6.90 \times 10^{3}}$
5	1.06×10^{3}	1.03×10^{3}	$\mathbf{1.01 \times 10^{3}}$	2.15×10^{3}	2.10×10^{3}	$\mathbf{2.04 \times 10^{3}}$
6	$\mathbf{3.47 \times 10^{2}}$	3.63×10^{2}	3.81×10^{2}	1.05×10^{3}	$\mathbf{8.51 \times 10^{2}}$	8.71×10^{2}
7	1.09×10^{2}	6.08×10^{1}	$\mathbf{4.02 \times 10^{1}}$	4.72×10^{3}	4.38×10^{2}	$\mathbf{2.40 \times 10^{2}}$
8	4.20×10^{3}	$\mathbf{8.58 \times 10^{2}}$	1.89×10^{3}	1.03×10^{4}	$\mathbf{2.27 \times 10^{3}}$	5.52×10^{3}
9	2.75×10^{3}	$\mathbf{2.28 \times 10^{3}}$	2.49×10^{3}	1.10×10^{4}	$\mathbf{5.64 \times 10^{3}}$	$\mathbf{5.64 \times 10^{3}}$
10	1.48×10^{5}	$\mathbf{1.51 \times 10^{4}}$	7.05×10^{4}	3.41×10^{5}	$\mathbf{4.24 \times 10^{4}}$	1.82×10^{5}
11	1.53×10^{5}	$\mathbf{6.04 \times 10^{4}}$	6.17×10^{4}	3.52×10^{5}	1.43×10^{5}	$\mathbf{1.34 \times 10^{5}}$
12	3.87×10^{2}	6.90×10^{1}	$\mathbf{4.90 \times 10^{1}}$	1.13×10^{3}	2.26×10^{2}	$\mathbf{1.38 \times 10^{2}}$
13	$\mathbf{2.17 \times 10^{1}}$	3.78×10^{1}	3.36×10^{1}	1.85×10^{2}	9.09×10^{1}	$\mathbf{7.76 \times 10^{1}}$
14	3.62×10^{1}	3.41×10^{1}	$\mathbf{3.02 \times 10^{1}}$	1.05×10^{3}	9.41×10^{1}	$\mathbf{6.67 \times 10^{1}}$
15	5.78×10^{2}	5.95×10^{2}	$\mathbf{5.37 \times 10^{2}}$	7.29×10^{4}	2.90×10^{4}	$\mathbf{8.57 \times 10^{3}}$
16	2.38×10^{2}	$\mathbf{2.04 \times 10^{2}}$	$\mathbf{2.04 \times 10^{2}}$	4.87×10^{2}	4.30×10^{2}	$\mathbf{4.21 \times 10^{2}}$
17	$\mathbf{3.17 \times 10^{7}}$	5.43×10^{7}	4.84×10^{7}	$\mathbf{7.27 \times 10^{7}}$	2.82×10^{8}	1.94×10^{8}
18	$\mathbf{3.82 \times 10^{5}}$	5.84×10^{5}	5.29×10^{5}	1.63×10^{7}	1.39×10^{6}	$\mathbf{1.10 \times 10^{6}}$
19	3.12×10^{2}	$\mathbf{2.02 \times 10^{2}}$	4.12×10^{2}	3.31×10^{3}	$\mathbf{6.01 \times 10^{2}}$	1.82×10^{3}
20	6.65×10^{5}	4.80×10^{5}	$\mathbf{3.63 \times 10^{5}}$	1.84×10^{6}	1.31×10^{6}	$\mathbf{7.69 \times 10^{5}}$
21	$\mathbf{2.28 \times 10^{7}}$	7.69×10^{7}	9.02×10^{7}	$\mathbf{2.10 \times 10^{8}}$	2.69×10^{8}	2.78×10^{8}
22	2.99×10^{4}	$\mathbf{1.13 \times 10^{4}}$	1.56×10^{4}	6.81×10^{4}	$\mathbf{3.00 \times 10^{4}}$	3.78×10^{4}
23	$\mathbf{1.70 \times 10^{4}}$	1.73×10^{4}	1.71×10^{4}	3.88×10^{4}	3.62×10^{4}	$\mathbf{3.56 \times 10^{4}}$
24	1.02×10^{4}	$\mathbf{8.79 \times 10^{3}}$	$\mathbf{8.79 \times 10^{3}}$	2.08×10^{4}	$\mathbf{1.88 \times 10^{4}}$	$\mathbf{1.88 \times 10^{4}}$
25	$\mathbf{1.00 \times 10^{4}}$	$\mathbf{1.00 \times 10^{4}}$	$\mathbf{1.00 \times 10^{4}}$	$\mathbf{2.01 \times 10^{4}}$	2.02×10^{4}	2.02×10^{4}
26	1.00×10^{4}	5.06×10^{3}	$\mathbf{5.03 \times 10^{3}}$	2.00×10^{4}	$\mathbf{1.04 \times 10^{4}}$	1.06×10^{4}
27	2.17×10^{4}	2.01×10^{4}	$\mathbf{1.93 \times 10^{4}}$	5.61×10^{4}	$\mathbf{4.56 \times 10^{4}}$	4.61×10^{4}
28	3.76×10^{4}	$\mathbf{3.49 \times 10^{4}}$	3.57×10^{4}	1.55×10^{5}	$\mathbf{8.13 \times 10^{4}}$	8.28×10^{4}
29	1.86×10^{8}	2.80×10^{7}	$\mathbf{2.70 \times 10^{7}}$	4.32×10^{9}	1.04×10^{8}	$\mathbf{7.53 \times 10^{7}}$
30	6.70×10^{5}	4.01×10^{5}	$\mathbf{2.06 \times 10^{5}}$	1.45×10^{7}	4.53×10^{6}	$\mathbf{2.51 \times 10^{6}}$

doi:10.6342/NTU202600592

by smSoD is denoted as mSoD→smSoD, and smSoD followed by mSoD is denoted as smSoD→mSoD.

Table A.3: Mean error of four discretization strategies on decomposable linkage problems with unknown linkage information across subproblem sizes $k \in \{2,3,4,5\}$ and numbers of subproblems $m \in \{10,20,50,100,200\}$.

(a) $k = 2$ and $k = 3$

	$k=2$				$k=3$			
m	mSoD	smSoD	mSoD→smSoD	smSoD→mSoD	mSoD	smSoD	mSoD→smSoD	smSoD→mSoD
10	1.22×10^{2}	3.58×10^{1}	1.01×10^{2}	$\mathbf{3.46\times10^{1}}$	1.27×10^{2}	$\mathbf{5.47\times10^{1}}$	2.89×10^{2}	2.75×10^{2}
20	3.29×10^{2}	1.18×10^{2}	6.29×10^{2}	$\mathbf{7.50\times10^{1}}$	8.79×10^{2}	3.50×10^{2}	1.36×10^{3}	$\mathbf{2.41\times10^{2}}$
50	1.02×10^{3}	$\mathbf{5.57\times10^{1}}$	8.92×10^{2}	1.26×10^{2}	2.16×10^{3}	3.80×10^{3}	1.62×10^{4}	$\mathbf{8.35\times10^{2}}$
100	6.84×10^{2}	6.92×10^{2}	9.92×10^{3}	$\mathbf{1.99\times10^{2}}$	4.59×10^{4}	6.48×10^{4}	1.38×10^{5}	$\mathbf{1.46\times10^{4}}$
200	2.34×10^{4}	5.79×10^{4}	1.30×10^{5}	$\mathbf{4.65\times10^{3}}$	3.68×10^{5}	5.20×10^{5}	6.74×10^{5}	$\mathbf{2.44\times10^{5}}$

(b) $k = 4$ and $k = 5$

	$k=4$				$k=5$			
m	mSoD	smSoD	mSoD→smSoD	smSoD→mSoD	mSoD	smSoD	mSoD→smSoD	smSoD→mSoD
10	4.75×10^{2}	3.51×10^{2}	1.43×10^{3}	$\mathbf{2.16\times10^{2}}$	1.66×10^{3}	$\mathbf{9.13\times10^{2}}$	2.66×10^{3}	9.68×10^{2}
20	$\mathbf{5.81\times10^{2}}$	1.99×10^{3}	3.89×10^{3}	9.10×10^{2}	7.19×10^{3}	8.77×10^{3}	1.75×10^{4}	$\mathbf{2.19\times10^{3}}$
50	2.94×10^{4}	3.45×10^{4}	7.09×10^{4}	$\mathbf{8.51\times10^{3}}$	8.52×10^{4}	7.90×10^{4}	1.30×10^{5}	$\mathbf{4.05\times10^{4}}$
100	1.93×10^{5}	2.16×10^{5}	3.14×10^{5}	$\mathbf{1.16\times10^{5}}$	3.10×10^{5}	3.22×10^{5}	4.55×10^{5}	$\mathbf{2.24\times10^{5}}$
200	7.91×10^{5}	9.12×10^{5}	1.14×10^{6}	$\mathbf{5.95\times10^{5}}$	1.06×10^{6}	1.21×10^{6}	1.52×10^{6}	$\mathbf{7.94\times10^{5}}$

doi:10.6342/NTU202600592

Table A.4: Mean error of four discretization strategies on the extended CEC 2014 benchmark with unknown linkage information across numbers of subproblems $m \in \{10, 20, 50, 100\}$.

	$m = 10$				$m = 20$			
Function	mSoD	smSoD	mSoD→smSoD	smSoD→mSoD	mSoD	smSoD	mSoD→smSoD	smSoD→mSoD
1	1.36×10^{8}	9.30×10^{7}	1.75×10^{8}	$\mathbf{8.83 \times 10^{7}}$	4.50×10^{8}	3.41×10^{8}	4.97×10^{8}	$\mathbf{2.82 \times 10^{8}}$
2	4.63×10^{7}	3.90×10^{5}	4.92×10^{6}	$\mathbf{2.40 \times 10^{5}}$	3.23×10^{7}	$\mathbf{4.63 \times 10^{4}}$	4.99×10^{8}	5.09×10^{4}
3	2.82×10^{5}	1.03×10^{5}	3.02×10^{5}	$\mathbf{7.45 \times 10^{4}}$	8.65×10^{5}	2.59×10^{5}	7.32×10^{5}	$\mathbf{2.22 \times 10^{5}}$
4	$\mathbf{2.59 \times 10^{2}}$	2.76×10^{2}	2.86×10^{2}	2.69×10^{2}	$\mathbf{6.55 \times 10^{2}}$	7.68×10^{2}	7.72×10^{2}	7.21×10^{2}
5	$\mathbf{1.99 \times 10^{2}}$	2.00×10^{2}	2.00×10^{2}	2.00×10^{2}	4.02×10^{2}	4.00×10^{2}	4.00×10^{2}	$\mathbf{4.00 \times 10^{2}}$
6	5.76×10^{1}	4.52×10^{1}	6.58×10^{1}	$\mathbf{4.51 \times 10^{1}}$	1.36×10^{2}	$\mathbf{1.10 \times 10^{2}}$	1.69×10^{2}	1.11×10^{2}
7	4.62×10^{0}	9.92×10^{0}	$\mathbf{4.20 \times 10^{0}}$	1.41×10^{1}	1.04×10^{1}	$\mathbf{5.91 \times 10^{0}}$	1.81×10^{1}	6.71×10^{0}
8	1.52×10^{2}	1.07×10^{2}	2.33×10^{2}	$\mathbf{1.06 \times 10^{2}}$	4.45×10^{2}	$\mathbf{2.76 \times 10^{2}}$	6.07×10^{2}	2.78×10^{2}
9	3.85×10^{2}	$\mathbf{1.44 \times 10^{2}}$	3.94×10^{2}	1.52×10^{2}	8.61×10^{2}	3.76×10^{2}	9.54×10^{2}	$\mathbf{3.76 \times 10^{2}}$
10	$\mathbf{3.29 \times 10^{3}}$	3.95×10^{3}	5.46×10^{3}	4.21×10^{3}	$\mathbf{8.81 \times 10^{3}}$	1.19×10^{4}	1.48×10^{4}	9.95×10^{3}
11	1.06×10^{4}	$\mathbf{9.81 \times 10^{3}}$	1.13×10^{4}	1.01×10^{4}	2.30×10^{4}	$\mathbf{2.23 \times 10^{4}}$	2.63×10^{4}	2.25×10^{4}
12	$\mathbf{4.67 \times 10^{0}}$	5.34×10^{0}	7.45×10^{0}	5.10×10^{0}	1.28×10^{1}	1.27×10^{1}	1.88×10^{1}	$\mathbf{1.21 \times 10^{1}}$
13	6.84×10^{0}	$\mathbf{2.91 \times 10^{0}}$	7.52×10^{0}	3.02×10^{0}	1.55×10^{1}	9.94×10^{0}	2.53×10^{1}	$\mathbf{9.59 \times 10^{0}}$
14	6.82×10^{0}	$\mathbf{4.78 \times 10^{0}}$	6.03×10^{0}	4.84×10^{0}	1.28×10^{1}	1.13×10^{1}	1.70×10^{1}	$\mathbf{1.06 \times 10^{1}}$
15	$\mathbf{4.44 \times 10^{1}}$	7.66×10^{1}	1.13×10^{2}	7.06×10^{1}	5.25×10^{2}	1.96×10^{2}	3.54×10^{3}	$\mathbf{1.95 \times 10^{2}}$
16	$\mathbf{3.94 \times 10^{1}}$	4.06×10^{1}	4.08×10^{1}	4.08×10^{1}	$\mathbf{8.41 \times 10^{1}}$	8.50×10^{1}	8.64×10^{1}	8.51×10^{1}
17	9.94×10^{6}	8.12×10^{6}	1.22×10^{7}	$\mathbf{8.05 \times 10^{6}}$	5.48×10^{7}	2.53×10^{7}	8.68×10^{7}	$\mathbf{1.82 \times 10^{7}}$
18	7.42×10^{5}	$\mathbf{8.50 \times 10^{4}}$	1.00×10^{7}	8.53×10^{4}	3.65×10^{6}	$\mathbf{1.71 \times 10^{5}}$	1.16×10^{7}	1.79×10^{5}
19	3.80×10^{1}	3.12×10^{1}	4.35×10^{1}	$\mathbf{2.94 \times 10^{1}}$	8.83×10^{1}	8.00×10^{1}	1.08×10^{2}	$\mathbf{6.99 \times 10^{1}}$
20	1.25×10^{5}	8.91×10^{4}	2.09×10^{5}	$\mathbf{6.55 \times 10^{4}}$	6.39×10^{5}	2.14×10^{5}	1.01×10^{6}	$\mathbf{1.77 \times 10^{5}}$
21	1.15×10^{7}	1.24×10^{7}	1.70×10^{7}	$\mathbf{1.06 \times 10^{7}}$	5.82×10^{7}	3.88×10^{7}	6.66×10^{7}	$\mathbf{3.69 \times 10^{7}}$
22	1.65×10^{3}	$\mathbf{1.47 \times 10^{3}}$	1.96×10^{3}	1.49×10^{3}	$\mathbf{3.86 \times 10^{3}}$	4.00×10^{3}	4.60×10^{3}	3.98×10^{3}
23	3.31×10^{3}	$\mathbf{3.29 \times 10^{3}}$	3.33×10^{3}	3.29×10^{3}	6.66×10^{3}	6.59×10^{3}	6.76×10^{3}	$\mathbf{6.59 \times 10^{3}}$
24	$\mathbf{1.71 \times 10^{3}}$	1.71×10^{3}	1.71×10^{3}	1.71×10^{3}	3.57×10^{3}	$\mathbf{3.52 \times 10^{3}}$	3.65×10^{3}	3.56×10^{3}
25	1.99×10^{3}	$\mathbf{1.96 \times 10^{3}}$	1.97×10^{3}	1.98×10^{3}	3.97×10^{3}	3.95×10^{3}	4.02×10^{3}	$\mathbf{3.94 \times 10^{3}}$
26	$\mathbf{1.02 \times 10^{3}}$	1.08×10^{3}	$\mathbf{1.02 \times 10^{3}}$	1.06×10^{3}	2.14×10^{3}	2.44×10^{3}	$\mathbf{2.14 \times 10^{3}}$	2.51×10^{3}
27	$\mathbf{4.07 \times 10^{3}}$	4.26×10^{3}	4.24×10^{3}	4.18×10^{3}	$\mathbf{8.51 \times 10^{3}}$	9.32×10^{3}	9.54×10^{3}	9.25×10^{3}
28	5.68×10^{3}	5.94×10^{3}	$\mathbf{5.65 \times 10^{3}}$	5.97×10^{3}	$\mathbf{1.21 \times 10^{4}}$	1.38×10^{4}	1.37×10^{4}	1.40×10^{4}
29	4.83×10^{6}	$\mathbf{3.99 \times 10^{6}}$	4.40×10^{6}	4.07×10^{6}	9.74×10^{6}	9.95×10^{6}	1.22×10^{7}	$\mathbf{9.61 \times 10^{6}}$
30	1.76×10^{4}	1.67×10^{4}	2.11×10^{4}	$\mathbf{1.63 \times 10^{4}}$	6.92×10^{4}	4.81×10^{4}	1.01×10^{5}	$\mathbf{4.20 \times 10^{4}}$

doi:10.6342/NTU202600592

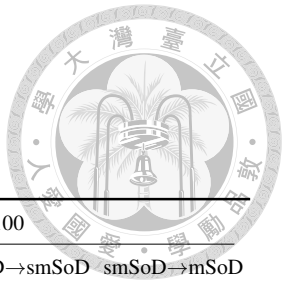


Table A.4: (continued)

	$m = 50$				$m = 100$			
Function	mSoD	smSoD	mSoD→smSoD	smSoD→mSoD	mSoD	smSoD	mSoD→smSoD	smSoD→mSoD
1	1.64×10^{9}	1.29×10^{9}	1.61×10^{9}	$\mathbf{1.06 \times 10^{9}}$	3.84×10^{9}	3.57×10^{9}	4.04×10^{9}	$\mathbf{2.86 \times 10^{9}}$
2	2.84×10^{7}	$\mathbf{5.97 \times 10^{5}}$	4.03×10^{9}	1.30×10^{6}	7.18×10^{9}	3.24×10^{10}	5.22×10^{10}	$\mathbf{2.53 \times 10^{9}}$
3	3.11×10^{6}	$\mathbf{1.28 \times 10^{6}}$	2.54×10^{6}	1.30×10^{6}	6.85×10^{6}	$\mathbf{3.68 \times 10^{6}}$	6.11×10^{6}	3.87×10^{6}
4	$\mathbf{2.42 \times 10^{3}}$	2.93×10^{3}	3.22×10^{3}	2.48×10^{3}	$\mathbf{7.90 \times 10^{3}}$	1.85×10^{4}	1.38×10^{4}	9.06×10^{3}
5	1.02×10^{3}	$\mathbf{1.00 \times 10^{3}}$	$\mathbf{1.00 \times 10^{3}}$	$\mathbf{1.00 \times 10^{3}}$	2.07×10^{3}	$\mathbf{2.00 \times 10^{3}}$	2.01×10^{3}	2.02×10^{3}
6	3.94×10^{2}	3.82×10^{2}	4.84×10^{2}	$\mathbf{3.61 \times 10^{2}}$	8.90×10^{2}	9.29×10^{2}	1.03×10^{3}	$\mathbf{8.87 \times 10^{2}}$
7	5.24×10^{1}	$\mathbf{4.58 \times 10^{1}}$	1.23×10^{2}	5.47×10^{1}	$\mathbf{2.49 \times 10^{2}}$	1.05×10^{3}	1.08×10^{3}	4.51×10^{2}
8	1.43×10^{3}	1.06×10^{3}	2.09×10^{3}	$\mathbf{9.71 \times 10^{2}}$	3.58×10^{3}	3.10×10^{3}	4.83×10^{3}	$\mathbf{2.82 \times 10^{3}}$
9	2.37×10^{3}	1.43×10^{3}	3.10×10^{3}	$\mathbf{1.42 \times 10^{3}}$	5.64×10^{3}	4.12×10^{3}	7.02×10^{3}	$\mathbf{3.83 \times 10^{3}}$
10	3.52×10^{4}	4.07×10^{4}	4.74×10^{4}	$\mathbf{3.46 \times 10^{4}}$	9.19×10^{4}	1.02×10^{5}	1.07×10^{5}	$\mathbf{9.03 \times 10^{4}}$
11	6.36×10^{4}	6.28×10^{4}	6.95×10^{4}	$\mathbf{6.03 \times 10^{4}}$	1.43×10^{5}	1.38×10^{5}	1.49×10^{5}	$\mathbf{1.33 \times 10^{5}}$
12	5.15×10^{1}	$\mathbf{4.53 \times 10^{1}}$	5.56×10^{1}	4.75×10^{1}	1.65×10^{2}	$\mathbf{1.20 \times 10^{2}}$	1.35×10^{2}	1.37×10^{2}
13	$\mathbf{4.87 \times 10^{1}}$	5.53×10^{1}	7.64×10^{1}	5.28×10^{1}	$\mathbf{1.30 \times 10^{2}}$	1.84×10^{2}	1.91×10^{2}	1.61×10^{2}
14	3.77×10^{1}	5.10×10^{1}	7.08×10^{1}	$\mathbf{3.21 \times 10^{1}}$	$\mathbf{1.22 \times 10^{2}}$	2.77×10^{2}	3.33×10^{2}	1.37×10^{2}
15	$\mathbf{8.32 \times 10^{2}}$	1.85×10^{3}	2.75×10^{4}	9.83×10^{2}	$\mathbf{1.86 \times 10^{4}}$	1.74×10^{5}	4.20×10^{5}	1.92×10^{4}
16	2.20×10^{2}	2.20×10^{2}	2.24×10^{2}	$\mathbf{2.18 \times 10^{2}}$	4.53×10^{2}	4.51×10^{2}	4.54×10^{2}	$\mathbf{4.47 \times 10^{2}}$
17	6.85×10^{8}	2.67×10^{8}	7.01×10^{8}	$\mathbf{1.39 \times 10^{8}}$	2.43×10^{9}	1.07×10^{9}	2.19×10^{9}	$\mathbf{6.31 \times 10^{8}}$
18	1.08×10^{7}	4.78×10^{5}	7.76×10^{6}	$\mathbf{4.70 \times 10^{5}}$	1.08×10^{7}	$\mathbf{9.69 \times 10^{5}}$	1.07×10^{8}	1.02×10^{6}
19	2.98×10^{2}	3.31×10^{2}	3.58×10^{2}	$\mathbf{2.55 \times 10^{2}}$	$\mathbf{7.38 \times 10^{2}}$	1.68×10^{3}	1.31×10^{3}	9.38×10^{2}
20	8.51×10^{6}	1.37×10^{6}	6.60×10^{6}	$\mathbf{1.31 \times 10^{6}}$	5.59×10^{7}	1.15×10^{7}	3.23×10^{7}	$\mathbf{7.65 \times 10^{6}}$
21	1.97×10^{8}	1.51×10^{8}	1.86×10^{8}	$\mathbf{1.21 \times 10^{8}}$	4.42×10^{8}	3.59×10^{8}	4.34×10^{8}	$\mathbf{3.18 \times 10^{8}}$
22	$\mathbf{1.22 \times 10^{4}}$	1.46×10^{4}	1.55×10^{4}	1.30×10^{4}	$\mathbf{2.94 \times 10^{4}}$	3.38×10^{4}	3.33×10^{4}	3.12×10^{4}
23	1.71×10^{4}	1.67×10^{4}	1.73×10^{4}	$\mathbf{1.66 \times 10^{4}}$	3.53×10^{4}	3.50×10^{4}	3.62×10^{4}	$\mathbf{3.41 \times 10^{4}}$
24	$\mathbf{9.27 \times 10^{3}}$	9.41×10^{3}	9.95×10^{3}	9.34×10^{3}	$\mathbf{1.97 \times 10^{4}}$	2.01×10^{4}	2.10×10^{4}	1.97×10^{4}
25	1.00×10^{4}	9.98×10^{3}	1.01×10^{4}	$\mathbf{9.94 \times 10^{3}}$	2.03×10^{4}	2.03×10^{4}	2.05×10^{4}	$\mathbf{2.01 \times 10^{4}}$
26	$\mathbf{5.83 \times 10^{3}}$	6.91×10^{3}	6.40×10^{3}	6.91×10^{3}	$\mathbf{1.31 \times 10^{4}}$	1.47×10^{4}	1.45×10^{4}	1.47×10^{4}
27	$\mathbf{2.40 \times 10^{4}}$	2.60×10^{4}	2.67×10^{4}	2.53×10^{4}	$\mathbf{5.20 \times 10^{4}}$	5.55×10^{4}	5.57×10^{4}	5.32×10^{4}
28	$\mathbf{3.55 \times 10^{4}}$	4.30×10^{4}	4.76×10^{4}	4.37×10^{4}	$\mathbf{8.36 \times 10^{4}}$	1.10×10^{5}	1.10×10^{5}	1.06×10^{5}
29	$\mathbf{3.16 \times 10^{7}}$	4.05×10^{7}	5.26×10^{7}	3.45×10^{7}	$\mathbf{1.24 \times 10^{8}}$	3.05×10^{8}	2.01×10^{8}	1.67×10^{8}
30	8.60×10^{5}	5.79×10^{5}	9.14×10^{5}	$\mathbf{3.24 \times 10^{5}}$	5.23×10^{6}	4.47×10^{6}	5.39×10^{6}	$\mathbf{2.33 \times 10^{6}}$

Table A.5: Mean error comparison between L-SHADE and smSoD→mSoD + ECGA on decomposable linkage problems with unknown linkage information across subproblem sizes $k \in \{2,3,4,5\}$ and numbers of subproblems $m \in \{10,20,50,100,200\}$. In this table, smSoD→mSoD + ECGA is denoted as smSoD→mSoD.

m	$k = 2$		$k = 3$		$k = 4$		$k = 5$	
	L-SHADE	smSoD→mSoD	L-SHADE	smSoD→mSoD	L-SHADE	smSoD→mSoD	L-SHADE	smSoD→mSoD
10	$\mathbf{0.00 \times 10^{0}}$	3.46×10^{1}	$\mathbf{0.00 \times 10^{0}}$	2.75×10^{2}	$\mathbf{0.00 \times 10^{0}}$	2.16×10^{2}	$\mathbf{8.51 \times 10^{-25}}$	9.68×10^{2}
20	$\mathbf{0.00 \times 10^{0}}$	7.50×10^{1}	$\mathbf{3.29 \times 10^{-25}}$	2.41×10^{2}	$\mathbf{6.96 \times 10^{-10}}$	9.10×10^{2}	$\mathbf{2.07 \times 10^{-3}}$	2.19×10^{3}
50	$\mathbf{1.59 \times 10^{-12}}$	1.26×10^{2}	$\mathbf{1.16 \times 10^{-1}}$	8.35×10^{2}	$\mathbf{3.07 \times 10^{2}}$	8.51×10^{3}	$\mathbf{7.64 \times 10^{3}}$	4.05×10^{4}
100	$\mathbf{1.09 \times 10^{0}}$	1.99×10^{2}	$\mathbf{6.18 \times 10^{3}}$	1.46×10^{4}	1.17×10^{5}	$\mathbf{1.16 \times 10^{5}}$	2.88×10^{5}	$\mathbf{2.24 \times 10^{5}}$
200	1.82×10^{4}	$\mathbf{4.65 \times 10^{3}}$	5.35×10^{5}	$\mathbf{2.44 \times 10^{5}}$	1.00×10^{6}	$\mathbf{5.95 \times 10^{5}}$	1.28×10^{6}	$\mathbf{7.94 \times 10^{5}}$

doi:10.6342/NTU202600592

Table A.6: Mean error comparison between L-SHADE and smSoD→mSoD + ECGA on the extended CEC 2014 benchmark with unknown linkage information across numbers of subproblems $m \in \{10, 20, 50, 100\}$. In this table, smSoD→mSoD + ECGA is denoted as smSoD→mSoD.

	$m = 10$		$m = 20$		$m = 50$		$m = 100$	
Function	L-SHADE	smSoD→mSoD	L-SHADE	smSoD→mSoD	L-SHADE	smSoD→mSoD	L-SHADE	smSoD→mSoD
1	$\mathbf{2.37 \times 10^{5}}$	8.83×10^{7}	$\mathbf{1.63 \times 10^{7}}$	2.82×10^{8}	1.11×10^{9}	$\mathbf{1.06 \times 10^{9}}$	4.51×10^{9}	$\mathbf{2.86 \times 10^{9}}$
2	$\mathbf{7.19 \times 10^{2}}$	2.40×10^{5}	$\mathbf{7.06 \times 10^{3}}$	5.09×10^{4}	1.59×10^{9}	$\mathbf{1.30 \times 10^{6}}$	2.03×10^{11}	$\mathbf{2.53 \times 10^{9}}$
3	$\mathbf{1.15 \times 10^{2}}$	7.45×10^{4}	$\mathbf{1.02 \times 10^{5}}$	2.22×10^{5}	$\mathbf{8.91 \times 10^{5}}$	1.30×10^{6}	$\mathbf{1.87 \times 10^{6}}$	3.87×10^{6}
4	3.43×10^{2}	$\mathbf{2.69 \times 10^{2}}$	$\mathbf{6.62 \times 10^{2}}$	7.21×10^{2}	2.60×10^{3}	$\mathbf{2.48 \times 10^{3}}$	3.10×10^{4}	$\mathbf{9.06 \times 10^{3}}$
5	2.02×10^{2}	$\mathbf{2.00 \times 10^{2}}$	4.15×10^{2}	$\mathbf{4.00 \times 10^{2}}$	1.06×10^{3}	$\mathbf{1.00 \times 10^{3}}$	2.15×10^{3}	$\mathbf{2.02 \times 10^{3}}$
6	$\mathbf{1.32 \times 10^{1}}$	4.51×10^{1}	$\mathbf{5.64 \times 10^{1}}$	1.11×10^{2}	$\mathbf{3.47 \times 10^{2}}$	3.61×10^{2}	1.05×10^{3}	$\mathbf{8.87 \times 10^{2}}$
7	$\mathbf{1.65 \times 10^{-1}}$	1.41×10^{1}	$\mathbf{4.14 \times 10^{0}}$	6.71×10^{0}	1.09×10^{2}	$\mathbf{5.47 \times 10^{1}}$	4.72×10^{3}	$\mathbf{4.51 \times 10^{2}}$
8	$\mathbf{5.60 \times 10^{1}}$	1.06×10^{2}	7.75×10^{2}	$\mathbf{2.78 \times 10^{2}}$	4.20×10^{3}	$\mathbf{9.71 \times 10^{2}}$	1.03×10^{4}	$\mathbf{2.82 \times 10^{3}}$
9	$\mathbf{1.31 \times 10^{2}}$	1.52×10^{2}	$\mathbf{2.90 \times 10^{2}}$	3.76×10^{2}	2.75×10^{3}	$\mathbf{1.42 \times 10^{3}}$	1.10×10^{4}	$\mathbf{3.83 \times 10^{3}}$
10	$\mathbf{2.68 \times 10^{3}}$	4.21×10^{3}	3.24×10^{4}	$\mathbf{9.95 \times 10^{3}}$	1.48×10^{5}	$\mathbf{3.46 \times 10^{4}}$	3.41×10^{5}	$\mathbf{9.03 \times 10^{4}}$
11	1.22×10^{4}	$\mathbf{1.01 \times 10^{4}}$	4.13×10^{4}	$\mathbf{2.25 \times 10^{4}}$	1.53×10^{5}	$\mathbf{6.03 \times 10^{4}}$	3.52×10^{5}	$\mathbf{1.33 \times 10^{5}}$
12	1.33×10^{1}	$\mathbf{5.10 \times 10^{0}}$	6.15×10^{1}	$\mathbf{1.21 \times 10^{1}}$	3.87×10^{2}	$\mathbf{4.75 \times 10^{1}}$	1.13×10^{3}	$\mathbf{1.37 \times 10^{2}}$
13	$\mathbf{4.22 \times 10^{-1}}$	3.02×10^{0}	$\mathbf{2.70 \times 10^{0}}$	9.59×10^{0}	$\mathbf{2.17 \times 10^{1}}$	5.28×10^{1}	1.85×10^{2}	$\mathbf{1.61 \times 10^{2}}$
14	4.90×10^{0}	$\mathbf{4.84 \times 10^{0}}$	$\mathbf{9.30 \times 10^{0}}$	1.06×10^{1}	3.62×10^{1}	$\mathbf{3.21 \times 10^{1}}$	1.05×10^{3}	$\mathbf{1.37 \times 10^{2}}$
15	$\mathbf{2.16 \times 10^{1}}$	7.06×10^{1}	$\mathbf{9.67 \times 10^{1}}$	1.95×10^{2}	$\mathbf{5.78 \times 10^{2}}$	9.83×10^{2}	7.29×10^{4}	$\mathbf{1.92 \times 10^{4}}$
16	$\mathbf{3.64 \times 10^{1}}$	4.08×10^{1}	8.61×10^{1}	$\mathbf{8.51 \times 10^{1}}$	2.38×10^{2}	$\mathbf{2.18 \times 10^{2}}$	4.87×10^{2}	$\mathbf{4.47 \times 10^{2}}$
17	$\mathbf{5.97 \times 10^{3}}$	8.05×10^{6}	$\mathbf{2.83 \times 10^{5}}$	1.82×10^{7}	$\mathbf{3.17 \times 10^{7}}$	1.39×10^{8}	$\mathbf{7.27 \times 10^{7}}$	6.31×10^{8}
18	$\mathbf{5.50 \times 10^{2}}$	8.53×10^{4}	$\mathbf{5.15 \times 10^{4}}$	1.79×10^{5}	$\mathbf{3.82 \times 10^{5}}$	4.70×10^{5}	1.63×10^{7}	$\mathbf{1.02 \times 10^{6}}$
19	4.10×10^{1}	$\mathbf{2.94 \times 10^{1}}$	8.33×10^{1}	$\mathbf{6.99 \times 10^{1}}$	3.12×10^{2}	$\mathbf{2.55 \times 10^{2}}$	3.31×10^{3}	$\mathbf{9.38 \times 10^{2}}$
20	$\mathbf{3.53 \times 10^{2}}$	6.55×10^{4}	$\mathbf{2.27 \times 10^{4}}$	1.77×10^{5}	$\mathbf{6.65 \times 10^{5}}$	1.31×10^{6}	$\mathbf{1.84 \times 10^{6}}$	7.65×10^{6}
21	$\mathbf{2.27 \times 10^{3}}$	1.06×10^{7}	$\mathbf{6.60 \times 10^{4}}$	3.69×10^{7}	$\mathbf{2.28 \times 10^{7}}$	1.21×10^{8}	$\mathbf{2.10 \times 10^{8}}$	3.18×10^{8}
22	$\mathbf{1.26 \times 10^{3}}$	1.49×10^{3}	7.59×10^{3}	$\mathbf{3.98 \times 10^{3}}$	2.99×10^{4}	$\mathbf{1.30 \times 10^{4}}$	6.81×10^{4}	$\mathbf{3.12 \times 10^{4}}$
23	$\mathbf{3.29 \times 10^{3}}$	3.29×10^{3}	$\mathbf{6.59 \times 10^{3}}$	6.59×10^{3}	1.70×10^{4}	$\mathbf{1.66 \times 10^{4}}$	3.88×10^{4}	$\mathbf{3.41 \times 10^{4}}$
24	$\mathbf{1.34 \times 10^{3}}$	1.71×10^{3}	3.90×10^{3}	$\mathbf{3.56 \times 10^{3}}$	1.02×10^{4}	$\mathbf{9.34 \times 10^{3}}$	2.08×10^{4}	$\mathbf{1.97 \times 10^{4}}$
25	2.00×10^{3}	$\mathbf{1.98 \times 10^{3}}$	3.99×10^{3}	$\mathbf{3.94 \times 10^{3}}$	1.00×10^{4}	$\mathbf{9.94 \times 10^{3}}$	2.01×10^{4}	$\mathbf{2.01 \times 10^{4}}$
26	$\mathbf{1.00 \times 10^{3}}$	1.06×10^{3}	$\mathbf{2.00 \times 10^{3}}$	2.51×10^{3}	1.00×10^{4}	$\mathbf{6.91 \times 10^{3}}$	2.00×10^{4}	$\mathbf{1.47 \times 10^{4}}$
27	$\mathbf{3.28 \times 10^{3}}$	4.18×10^{3}	$\mathbf{7.16 \times 10^{3}}$	9.25×10^{3}	$\mathbf{2.17 \times 10^{4}}$	2.53×10^{4}	5.61×10^{4}	$\mathbf{5.32 \times 10^{4}}$
28	$\mathbf{3.94 \times 10^{3}}$	5.97×10^{3}	$\mathbf{9.00 \times 10^{3}}$	1.40×10^{4}	$\mathbf{3.76 \times 10^{4}}$	4.37×10^{4}	1.55×10^{5}	$\mathbf{1.06 \times 10^{5}}$
29	$\mathbf{5.69 \times 10^{5}}$	4.07×10^{6}	9.88×10^{6}	$\mathbf{9.61 \times 10^{6}}$	1.86×10^{8}	$\mathbf{3.45 \times 10^{7}}$	4.32×10^{9}	$\mathbf{1.67 \times 10^{8}}$
30	$\mathbf{9.85 \times 10^{3}}$	1.63×10^{4}	$\mathbf{3.33 \times 10^{4}}$	4.20×10^{4}	6.70×10^{5}	$\mathbf{3.24 \times 10^{5}}$	1.45×10^{7}	$\mathbf{2.33 \times 10^{6}}$

doi:10.6342/NTU202600592