H -polytope decomposition-based algorithm for continuous optimization

Ghazaleh Khodabandelou, Amir Nakib

Version of Record: https://www.sciencedirect.com/science/article/pii/S0020025521000232
Manuscript_ca97e56d8ed748480c4bd36f58a27954

$\mathcal{H}$-Polytope Decomposition-based Algorithm for Continuous Optimization

Ghazaleh Khodabandelou[a,*], Amir Nakib[a]

[a]*Laboratory of Images, Signals and Intelligent Systems (LISSI), University of Paris-Est Creteil, France*

Abstract

This paper presents a new fractal search space decomposition-based algorithm to address the issue of scaling up the divide and conquer approach to deal with large scale problems (up to 50 continuous decision variables). The proposed algorithm, called polyFrac, fractally decomposes the search space using hyper-polytopes. It allows moving throughout different granularity levels by only computing the average of vertices of a hyper-polytope to obtain the coordinates of the centroids. Only the most promising hyper-polytopes are decomposed into child-polytopes. Then, a simple deterministic local search (single solution-based metaheuristic) is used to perform the intensification process to find the best solution within the selected lowest hyper-polytope. The proposed algorithm performance is evaluated on the well-known SOCO 2011, CEC 2013, and CEC 2017 benchmarks and compared with 26 states of the art algorithms. A real-world optimization problem is also used to calibrate its performance. The obtained results show that polyFrac outperforms all the algorithms. Moreover, experimental results and analysis suggest that polyFrac is a highly competitive optimization algorithm for solving large-scale and complex optimization problems.

Keywords: Continuous optimization, Metaheuristic, Multi-objective, Deep learning, Fractals, Large-scale optimization.

1. Introduction

In recent years, the emergence of supercomputers and cloud infrastructure has greatly increased the possibility of dealing with increasingly complex problems (multi-modality, non-linearities, uncertainties in parameters, etc.). Metaheuristics algorithms are widely used to deal with these complex optimization problems. Indeed, they address any optimization problem without any knowledge of the objective function. Especially, population-based metaheuristics are the most used to deal with large scale problems (dimension greater than 50) [1]. Metaheuristics can be categorized into non-nature-inspired and nature-inspired algorithms. Nature-inspired metaheuristic algorithms can be classified into five main categories: evolutionary-based, physics-based, chemistry-based, human-based, and swarm intelligence-based [2]. Evolutionary algorithms are inspired by biological evolution in nature using different operators such as mutation, crossover, selection, and reproduction to find better candidate solutions [3], [4]. For instance, evolutionary algorithms, evolution strategy, and differential evolution algorithms have been extensively modified and adapted for different issues. Those modifications increase the performances but also increase significantly the complexity of their implementation. Moreover, the stochastic nature of metaheuristics is a restrictive element when repeatability is crucial In these cases, metaheuristics allow enhancing the deterministic algorithm's parameters (*i.e.* tuning deterministic algorithms). Besides, state-of-the-art metaheuristics are not easy to implement and have a set of parameters that are not easy to fit. Furthermore, it should be noticed that local minima are frequent in small problems, but they are infrequent in large-scale cases, however, saddle points (points with zero gradient) are frequent (see [5] for further theoretical details). This observation highlights the difficulty to use gradient information when dealing with large-scale problems. This work deals with large-scale optimization problems and proposes an efficient low-complex and easy-to-implement single solution-based metaheuristic. The proposed algorithm, called polyFrac, is based on dividing fractally (dividing recursively) the search space. Herein, the fractal notion consists of the decomposition of the search space using the geometric fractals. There exist several patterns in nature approximating a

*Corresponding author
Email address: ghazaleh.khodabandelou@u-pec.fr (Ghazaleh Khodabandelou)

centroidal polytope tessellation such as the cornea cells, the causeway surface, and tilapia skin. In the case of decomposing the search space fractally using hyper-polytopes, it is obvious that an optimal solution can be found exhaustively by exploring all child-polytopes at the lower level. However, it is too time-consuming and resource exhaustive. To deal with this issue, an efficient heuristic is proposed to select the most promising region in the search space for further decomposition. The goal is to find the best solution inside a reduced sub-region. The proposed algorithm, called polyFrac, is composed of two heuristics are proposed: Promising polytope selection (exploration strategy) and Local Search (exploitation strategy). In the exploration phase, the most promising hyper-polytope is selected for further decomposition. In the exploitation phase, a local search is used for the child-polytopes of the lower level to find the best solution inside a reduced sub-region. Moreover, the hyper-polytopes' centroids can be computed analytically with no need to save the past positions. Then, only the best positions met are saved. The key contribution of the proposed approach consists of dividing the search space recursively into a subset of hyper-polytopes shapes as fractals (*i.e.* self-similar patterns) with different granularity levels. The levels are categorized from fine-grained to coarse-grained ones. These geometric shapes have the advantage of covering exactly the entire of an N-dimensional search space. Indeed, the proposed approach allows a recursive decomposition of the search space to a fixed number of non-overlapping regions. In this work, a centroidal polytope tessellation is used where the generated point for each polytope is also its centroid, *i.e.* the mass center. This property ensures that the optimum search is restricted inside a given hyper-polytope. Moreover, the proposed algorithm allows moving throughout different granularity levels *i.e.* from fine-grained to coarse-grained levels, and vice versa by computing the centroids' coordinates. The polytopes centroids can be easily computed by averaging a given polytope's vertices. This manner of computing the centroid allows preserving a low computation. It can be noticed that the proposed algorithm is neither a population-based metaheuristic nor stochastic, polyFrac is a single-solution deterministic algorithm. The advantages of the proposed approach that motivated this work are:

- The hyper-polytopes shapes allow covering the entire of an N-dimensional search space without overlapping regions.
- The centroidal polytope tessellation ensures that the optimum search is restricted inside a given polytope.
- The polytopes centroids can be readily computed. This property allows for preserving a low computation complexity.
- Besides, this strategy of decomposing is intrinsically parallel and allows running the algorithm on multi-threaded and multi-node environments.

The proposed algorithm performance is evaluated on large-scale optimization benchmark functions from SOCO 2011, CEC 2013, and CEC 2017 benchmarks and compared with 26 different algorithms for dimensions varying from $50D$ to $1000D$. A real-world optimization problem is also used to calibrate the algorithm performance. Compared with population-based metaheuristics, the polyFrac algorithm performance shows its efficiency in dealing with large-scale optimization problems.

The rest of the paper is organized as follows: Section 2 presents the related work. Section 3 exposes the polytope fractal decomposition principals. Section 4 details the proposed algorithm. Results and comparison with other approaches are reported and discussed in Section 5. Section 6 concludes the paper.

2. Related work

Few studies in the field have tackled the optimum search using the fractal concept. In what follows, we review the salient state of the art algorithms. Authors in [6] address the problem of bound-constrained global optimization by partitioning the objective function over multiple scales for the global optimum. DIRECT (DIviding RECTangles) algorithm was proposed in [7]. This algorithm overcomes the need to specify a Lipschitz constant issue by performing an iterative search to divide the search space into all possible hyperintervals.

In each iteration, the most promising hyperintervals are chosen for further dividing. Despite its tremendous success to solve optimal design issue, for dimension greater than 10 the algorithm performance drastically deteriorates in terms of computation time and solution quality. FRACTOP is a metaheuristic based on a geometric decomposition of the search space into hypercubes [8]. However, applied to high dimensional problems, the algorithm is computationally expensive since the number of vertices rises exponentially. The idea behind the decomposition process is that the subregions are visited only once. This property makes the algorithm advantageous for classical problems (dimension less than 10). Nevertheless, it is not scalable because

of its exponential complexity. Indeed, each region is decomposed into 2^n subregions where n stands for the problem dimension and a metaheuristic such as simulated annealing [9] was used to search for solutions at random. Another approach that uses the fractal geometry for evolutionary algorithms, called Multiple Optima Sierpinski Searcher, was proposed in [10]. This approach used the Sierpinski triangle as the fractal geometrical shape. The fractals are created using the chaos game with a regular triangle and the factor $1/2$. The approach uses $n + 1$ generators instead of 2^n generator samples to reduce the computational cost. However, this approach has high complexity and fails to cover all the feasible regions using fractals.

A recent approach [11] proposed a hyperspherical decomposition of the search space, called FDA. This approach performs well on large-scale continuous optimization problems with some theoretical convergence properties. However, it allows covering all the search space only in case of small dimension problems. Moreover, this approach uses the overlapping hyperspheres that introduced further constraints and complexity to the algorithms. The proposed approach tackles these by completely covering the entire search domain for large-scale and big optimization problems. Moreover, the FDA does not perform well in case of non-separable problems.

In the literature, several approaches propose a geometric decomposition of the search space using classic metaheuristics, such as differential evolution, particle swarm optimization, and genetic algorithms. Nevertheless, they fail to deal when the dimension of the problems increase. The island artificial bee colony (iABC) [12] is a version of artificial bee colony (ABC) algorithm [13] based on the island model concepts. The population concept is used in the algorithm's structure by applying the island model to enhance its diversity. The population is divided into a set of sub-populations, *i.e.* islands. A Multi-population algorithm is proposed in [14] for solving the constrained and unconstrained numerical and engineering optimization problems. It uses an adaptive concept for dividing the population into sub-populations. The similarity between these algorithms and polyFrac is in their theoretical characteristics, *i.e.* they are black-box optimization algorithms. Whereas all the three algorithms mentioned above are population-based algorithms, polyFrac is a single solution-based.

Furthermore, recent literature on metaheuristics sees hybridization with machine learning techniques attracting many researchers.

In [15], the authors proposed a hybridized monarch butterfly optimization algorithm using a convolutional neural network by performing a hybridization with two other swarm intelligence algorithms. A memetic particle swarm optimization (MPSO), the authors proposed to improve the local search ability of the standard particle swarm algorithm [16]. The proposed algorithm is tested with various unconstrained, constrained, min-max, and integer programming problems. A hybrid algorithm combining firefly and particle swarm optimization (HFPSO) was proposed in [17]. This algorithm takes benefit from the strong points of both particle swarm and firefly algorithm mechanisms by determining the start of the local search process using the previous global best fitness values. Proactive particles in swarm optimization (PPSO) [18] proposed an algorithm based on particle swarm optimization (PSO) using fuzzy logic. The proposed algorithm uses a self-tuning version strategy based on fuzzy logic to dynamically determine the best settings for the inertia weight, cognitive factor, and social factor. Authors in [19] proposed a hybridizing salp swarm algorithm with PSO algorithm (HSSAPSO). The proposed approach combined the salp swarm algorithm (SSA) and PSO to remove their drawbacks, such as the trapping in local optima and the unbalanced exploitation. HSSAPSO uses the PSO approach velocity phase to avoid the premature convergence of the optimal solutions in the search space, escape from ignoring local minima, and improve the exploitation tendencies.

Cooperative Co-evolution (CC) methods were introduced by Potter *et al.* [20] are based on decomposing a high-dimensional problem and tackling its subcomponents individually. However, this approach is not efficient on non-separable problems since the interdependencies among different variables could not be captured well enough by the algorithms. Potter's decomposition was included in a PSO algorithm [21] where two cooperative PSO models proposed. However, these two models were only tested on functions of up to 30 dimensions [21] and 190 dimensions [22].

In [23] Yang *et al.* suggest a decomposition strategy based on random grouping. Without prior knowledge of the non-separability of a problem, it has shown that random grouping increases the probability of two interacting variables being allocated to the same subcomponent, thereby making it possible to optimize these interacting variables in the same subcomponent rather than across different subcomponents. An adaptive weighting scheme also proposed to fine-tune the solutions [23]. Other decomposition strategies have been proposed in the differential evolution algorithms literature. A splitting-in-half strategy proposed by Shi *et al.* [24]

decomposed the search space into two subcomponents, each evolved by a separate subpopulation. However, this strategy does not scale up well and does not perform well when the problem dimension becomes very large. Yang *et al.* [23] proposed a decomposition strategy based on random variables grouping and applied it to a cooperative coevolution differential evolution on high-dimensional non-separable problems with up to 1000 continuous variables.

One of the most cited algorithms in continuous optimization is the CMA-ES algorithm. It tunes the mutation parameters through computing a covariance matrix and hence correlated step sizes in all dimensions [25], based on an adaptive procedure. However, most of the published results of CMA-ES were on functions of up to 100 dimensions [25], [26]. One major drawback of CMA-ES is its cost in calculating the covariance matrix, which has a complexity of $O(n^2)$. When the dimension increases, this cost rapidly rises. Since its first version, many variants were proposed in the literature to tackle its limitations. One of them, called RB-IPOP-CMAES, is chosen here to compare against the proposed algorithm.

3. Polytope Fractal Decomposition

We propose a geometrical fractal decomposition based on hyper-polytope form as the geometric object. The polytope is defined as an N-dimensional generalization of the 2-dimensional polygon or 3-dimensional polyhedron. Figure 1 depicts an example of a centroidal polytope fractal decomposition with a fractal dimension of six and three decomposition levels where the points represent the centroids of polytopes. The first step consists of fractal decomposition that is a recursive decomposition of the search space with a *fixed* number of centroidal polytopes at each level. The number of polytopes inside a reference polytope indicates the fractal dimension. The polytope shape represents several advantages compared to the other schemes, *e.g.* polytope, squares such as its ability to cover the entire search space or the low complexity in large-scale problems. A key characteristic that stems from these properties is scalability, which allows decomposing the whole search space using polytopes fractals. As mentioned in the previous section, other geometric shapes do not cover all the possible search space or are complex to implement since their complexity at the generalization to the N-dimensional increases exponentially.

In this work, the considered geometrical fractal decomposition forms are the regular and the irregular convex polytopes.

3.1. Polytope Tessellations

Definition of polytope tessellations. A polytope is a partition of the plane into n convex polytopes (in 2D). Each partition contains one seed such that every point in the partition is nearer to its own seed than any other.

Definition 1 (Polytope Tessellation)**.** *Given a set of points $\{z_i\}_{i=1}^{k}$ belonging to the closed set $\overline{\Omega} \in \mathbb{R}^N$, the polytope region $\hat{\mathcal{P}}_i$ corresponding to the point z_i is defined by:*

$$\hat{\mathcal{P}}_i = \{x \in \Omega \;\; \|x - z_i\| < \|x - z_j\| \;\; for j = 1, \cdots, k, j \neq i\}. \tag{1}$$

where polytope region is a convex area in a Euclidean space that contains every point nearest to a point belonging to a set of $\{z_i\}_{i=1}^{k}$ with regard to all the other points.

Definition 2 (Tessellation)**.** *Given an open set $\Omega \in \mathbb{R}^N$, the set $\{\mathcal{P}_i\}_{i=1}^{k}$ is called a tessellation of Ω if $\mathcal{P}_i \subset \Omega$ for $\{i = 1, \cdots, k\}, \mathcal{P}_i \cap \mathcal{P}_j = \oslash$ for $i \neq j$, and $\bigcup_{i=1}^{k} \overline{\mathcal{P}}_i = \overline{\Omega}$.*

3.2. Optimum condition for centroidal polytopes

The substantial restriction for the centroidal polytopes tessellation is that each polytope generator has to be the centroid for its related polytope region as depicted in Figure 2. Thus, the definition of a centroidal polytope is the aforementioned definition of polytope tessellations with an extra constraint on the location of the generators. Given the set of polytope regions $\{\mathcal{P}_i\}_{i=1}^{k}$, the mass centroid c_i over a region with probability density $\rho(x)$ is defined as:

$$c_i = \frac{\int \mathcal{P}_i x \rho(x)}{\sum_{\mathcal{P}_i} \rho(x)} \tag{2}$$

where the density function $\rho(x) \geq 0$ and x is a vector in $\mathbb{R}^N$. In addition, in order to considering the polytope as a centroidal polytope tessellation, given k generators $\{z_i\}_{i=1}^{k}$, the following condition must be respected:

$$z_i = c_i, \qquad i = 1, \cdots, k. \tag{3}$$

where z_i points that are used as generators for the polytope regions $\hat{\mathcal{P}}_i$ are in turn the mass centroids of those regions. Such a tessellation is called a centroidal polytope tessellation.

As shown in Figure 2, the centroidal polytope tessellation algorithm at the first step generates the irregular polytopes that converge towards regular ones with several iterations. The polytope regions correspond to 10 randomly selected points in the search space. After 12 iterations, the generators points (blue points) converge to the centroids of polytope tessellation (red points).

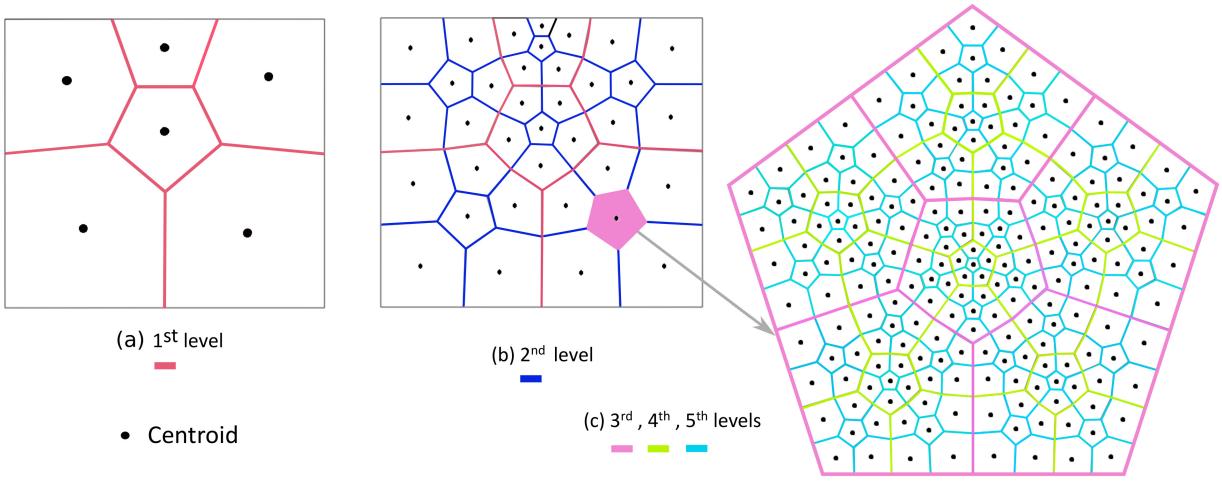

Figure 1: An overview of the polytopes fractal decomposition of the search space for a granularity decomposition level equal to 5.

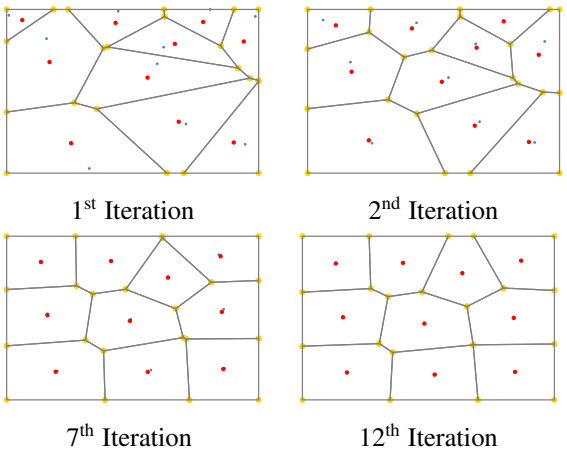

Figure 2: The polytope regions corresponding to 10 randomly selected points in the search space. The blue points are the polytope generators and the red points are the centroids of the corresponding polytope regions. After 12 iterations the generators points converge to the centroids of polytope tessellation.

3.3. Advantages of polytope decomposition

In what follows, the advantages of polytope decomposition compared to the hyperspherical or other similar forms of partitioning are given:

- These geometric shapes have the advantage of covering the entire of an N-dimensional search space without surpassing it or being inferior to it. Indeed, polyFrac proposes a recursive decomposition of the search space Ω to a fixed number of non-overlapping regions as portrayed in Figure 1 and defined in **Definition 2**.

  A hyperspherical form does not allow covering all the search space. An inflation rate proposed to overcome this problem by increasing the radius of each hypersphere [11]. It generates hyperspheres that overlap or exceed the search space. It should be noted that despite inflating hyperspheres, all search space is not fully covered. In both scenarios, a hyperspherical shape does not lead to an optimal solution. Furthermore, this property allows ignoring the whole part on the search space coverage proof, as it is done in the hyperspherical form (see [11]).

- In this study we use a centroidal polytope tessellation in which the generated point, *i.e.* the seed, for each polytope is also its centroid, *i.e.* the mass center. This property ensures that the optimum search is restricted inside a given polytope.

- The polytopes centroids are easily computed. Indeed, it is possible to move throughout different granularity levels, *i.e.* from fine-grained to coarse-grained levels and vice versa by calculating the average of vertices of a polytope to obtain the coordinates of the centroids. This manner of the centroid calculation allows for preserving a low computation complexity. One can see that this approach allows approximating polytopes centroids by several iterations as illustrated in Figure 2. However, an approximate center is sufficient for the polyFrac algorithm. Then, it is defined by:

$$c^{\mathcal{P}} = \frac{1}{k}\sum_{i=1}^{k} \mathcal{V}_i^{\mathcal{P}} \tag{4}$$

where $c^{\mathcal{P}}$ stands for the centroid of a given polytope $\mathcal{P}$ and $\mathcal{V}_i^{\mathcal{P}}$ is the vertices of the polytope with k as the number of polytopes centroid, which is equal to the number of decomposition in each level (a fixed value).

Lloyd's algorithm [27] allows determining the exact position of centroid with a recursive algorithm. However, this algorithm cannot be used for an N-dimensional search space.

4. Poly Fractal Decomposition Algorithm

The proposed algorithm, called polyFrac, searches for an optimal global solution (if it is known). The naive approach is to explore by brute-force search all fine-grained polytopes in the last level, called child-polytopes. However, this method is too time-consuming and resource-intensive. The polyFrac algorithm uses two heuristics to overcome this problem: (i) promising polytope selection heuristic, and (ii) local search heuristic: this heuristic looks for the best solution at the last (lowest) level.

4.1. Poly fractal algorithm

Algorithm 1 represents the proposed algorithm. As previously pointed out, it uses the polytope $\mathcal{H}$ to divide the search space iteratively into $n\times$ dimension child-polytopes $f_i^{\mathcal{P}}$ where $i = 1, \cdots, nD$, where n is a value to be determined. In this work, the value of n is set to 2.

The algorithm of polyFrac decomposes for each iteration the search space into a constant number $2 \times D$ of convex child-polytopes $\mathcal{P}^l$ at decomposition level l. The decomposition procedure iterates until the desired level l is reached. Then, the quality Q_i^l of each child-polytope is evaluated using the procedure described in Section 4.2. The child-polytopes are sorted with respect to the highest quality determining the next polytope to be decomposed. Therefore, the algorithm can direct the search for the most promising area and starts in the best position.

Two points x_1 and x_2 inside those subregions allow for determining the initial polytope subregions. Although these points can be obtained at random, it is recommended to calculate their coordinates to decrease the number of iterations and speed up convergence. The subregion of the search space limited by a child-polytope $f_i^{\mathcal{P}}$ is defined by:

$$f_i^{\mathcal{P}} = \{x \in \mathbb{R}^D : y = \vec{w}^T(x - x_0)\}, \quad with \quad i \in L^l$$
$$f_i^{\mathcal{P}} = \begin{cases} f_i^{\mathcal{P}(+)} & \text{y>0} \\ f_i^{\mathcal{P}(-)} & \text{y<0} \end{cases} \tag{5}$$

Algorithm 1: Poly-Fractal Decomposition Algorithm

Input: $\mathcal{V}_i$: vertices, k = 5: fractal depth,
$w_{min} = 1 \times e^{-20}$: tolerance threshold
$\phi = 0.5$: step-size reduction, D: dimension

$\mathcal{N}_{eval} = 1$: number of the objective function evaluations

$\vec{c}$: search space barycenter

$f(\vec{c})$: initializing the best fitness value for corresponding position $\vec{c}$

Initialize the level variable: $l = 1$

$F_{max} = 5000 \times D$: stopping criterion
while $\neq$ *Stopping criteria* **do**
  Decompose the current polytope $\mathcal{P} \rightarrow$ *Algorithm 2*
  for $2 \times D$ *l-level polytope* **do**
    Apply the promising polytope selection descried in equations (10), (11), (12)
    Sort the $2 \times D$ polytopes at the current l-level
    Replace the current polytope $\mathcal{P}$ by the first of the sorted polytopes at the current level
  if $l == k$ **then**
    for *Each* $2 \times D$ *of kth-level polytope* **do**
      Local search heuristics $\rightarrow$ *Algorithm 4*
      if $\neq F_{max}$ **then**
        Move-up $\rightarrow$ *Algorithm 3*
      else
        $l = l + 1$
  Result: the best solution and its position

$$
\begin{aligned}
&with \quad x_1 = \frac{v_1^{max} + c^{\mathcal{P}}}{2}, \quad x_2 = \frac{v_2^{max} + c^{\mathcal{P}}}{2}, \\
&x_0 = \frac{x_1 + x_2}{2} \\
&where \quad c^{\mathcal{P}} = \frac{\sum_i^n x_i}{n}, \quad \vec{w} = x_1 - x_2
\end{aligned} \tag{6}
$$

where v_1^{max} and v_2^{max} stand for the polytope vertices with maximum distances to the corresponding centroid $c^{\mathcal{P}}$. The child-polytope i center is denoted by x_0, and L^m is the set of indices that constitutes the polytope at level l, respectively. The center of the vector linking $x1$ and x_2 is represented by x_0, where they are perpendicular to the same plane $\vec{w}^T(x - x_0)$. The vertices on positive $\mathcal{V}^+{}_i$ and negative $\mathcal{V}^-{}_i$ sides of the plane are obtained by projecting vertices $\mathcal{V}_i$ on the plane $\vec{w}^T(x - x_0)$:

$$
\begin{aligned}
&y = \vec{w}^T(\mathcal{V}_i - x_0) = \begin{cases} \mathcal{V}_k^{(+)} & \text{y>0} \\ \mathcal{V}_j^{(-)} & \text{y<0} \end{cases} \\
&where \quad k \in \mathbb{R}^+, j \in \mathbb{R}^-, i \in \mathbb{R}
\end{aligned} \tag{7}
$$

The vertices $\mathcal{V}_k^{(+)}$ and $\mathcal{V}_j^{(-)}$ represent both sides of the plane y, they are not enough to generate child-polytopes covering the entire of the subregion. An appropriate child-polytope is calculated by the union of vertices u_i, situated on the plane separating positive and negative regions, with $\mathcal{V}_k^{(+)}$ on one side, and with $\mathcal{V}_j^{(-)}$ on the other side. The vertices u_i are defined as:

$$
\begin{aligned}
&u_i = \alpha * \mathcal{V}_k^{(+)} + (1 - \alpha) * \mathcal{V}_j^{(-)} \quad with \\
&where \quad \alpha = \frac{(x_0 - \mathcal{V}_i) * \vec{w}^T}{(\mathcal{V}_k^{(+)} - \mathcal{V}_j^{(-)}) * \vec{w}^T}, \quad 0 \leq \alpha \leq 1 \\
&\begin{cases} \mathcal{V}_i^{(1)} = \mathcal{V}_k^{(+)} \cup u_i & \text{y>0} \\ \mathcal{V}_i^{(2)} = \mathcal{V}_j^{(-)} \cup u_i & \text{y<0} \end{cases}
\end{aligned} \tag{8}
$$

The average of vertices $\mathcal{V}_i^{(1)}$ and $\mathcal{V}_i^{(2)}$ are calculated as the center of each new subregion:

$$
\begin{aligned}
c_1^{\mathcal{P}} &= \frac{\sum_i^n \mathcal{V}_i^{(1)}}{n} = x_1^{new} \\
c_2^{\mathcal{P}} &= \frac{\sum_i^n \mathcal{V}_i^{(2)}}{n} = x_2^{new}
\end{aligned} \tag{9}
$$

The new vertices x_1^{new} and x_2^{new} are used in Equation (6) to further decompose the child-polytopes. This procedure is iteratively repeated until the l-level is reached. At each iteration, all child-polytopes are sorted. Then the most promising one is chosen to be further decomposed. Algorithm 2 describes the polytope decomposition procedure. The process is iteratively repeated until the desired depth (D) is reached. Finally, the local search procedure (LS) is carried out to the sorted child-polytopes $f_i^{\mathcal{P}}$ (see Algorithm 3). Once the last level polytopes are entirety inspected (using a moving-up strategy), the search is raised to another part using the past depth. This is done by supplanting the current polytope $\mathcal{P}$ with the most promising child-polytopes $f_i^{\mathcal{P}}$ (determined using fitness value). The process stops when all the current level child-polytopes are visited, or the stopping criteria are reached.

The proposed approach has some similarities with the Depth-first branch-and-bound search ($B\&B$) [28] such as splitting the search space into sub-regions where each branch is a specific part of the entire solution. However, it differs from $B\&B$ by proposing an intensive search in the sense that each promising sub-region further decomposes and meticulously searched where each branch is a part of the search space. Indeed, the proposed method is a single solution-based metaheuristic. The number of polytopes visited measures diversity. The most promising region selection defines how diversification is controlled. The polyFrac compared to the depth-first branch and bound technique ($B\&B$), often used in combinatorial optimization, proposes each branch as a part of the search space rather than a singular part of the whole solution. Some areas are divided and, areas that do not seem to be hopeless are further investigated while the most promising one is searched more intensively.

4.2. *Promising polytope selection (Exploration strategy)*

The promising polytope selection heuristic aims to select the most potential coarse-grained polytope that could contain the global optimum (or at least the best solution). It evaluates each coarse-grained polytope p to decompose it into a constant number of fine-grained polytopes, *i.e.* child-polytopes. It should be noted that this procedure records a trace of the best solution (Sol^{best}) and its coordinates in the best position.

The polytope quality is assessed by two random points $\vec{b_1}$ and $\vec{b_2}$ generated using the following equations:

$$
\begin{aligned}
\vec{b_1}^{\mathcal{P}} &= c_l^{\mathcal{P}} + \frac{max(\mathcal{D}_l^{\mathcal{P}}(c_l^{\mathcal{P}}, u_i))}{\sqrt{D}} \\
\vec{b_2}^{\mathcal{P}} &= c_l^{\mathcal{P}} - \frac{max(\mathcal{D}_l^{\mathcal{P}}(c_l^{\mathcal{P}}, u_i))}{\sqrt{D}}
\end{aligned} \tag{10}
$$

where $c_l^{\mathcal{P}}$ represents the polytope center at level l. Then, fitnesses are calculated for positions of these three points $c_l^{\mathcal{P}}, \vec{b_1}, \vec{b_2}$ as $f_c^{\mathcal{P}}$ f_b^1, f_b^2. Finally, their corresponding distances (Euclidean) to the current best position $\mathcal{B}$

Algorithm 2: $\mathcal{H}$-Polytope Decomposition

Input: Vertices $\mathcal{V}_i \in \Omega$

for *$2 \times D$, 1-level polytope* **do**

Compute the polytope center x_c by averaging over vertices:

$x_c = \frac{\sum_i^n \mathcal{V}_i}{n}$

Compute the distance between x_c and all the polytope vertices:

$\mathcal{D}_1^{\mathcal{P}} = Dist(x_c, \mathcal{V}_i)$

Choose the vertex v_1 which has the maximum distance to x_c:

$\mathcal{D}_2^{\mathcal{P}} = Max(\mathcal{D}_{\mathcal{P}})$

Find vertex v_2 which has the maximum distance to v_1 among all other vertices

Compute two points x_1 and x_2 inside the 2 new polytopes:

$x_1 = \frac{v_1 + x_c}{2}$

$x_2 = \frac{v_2 + x_c}{2}$

Compute centroids x_0 of 2 new polytopes and vector $\vec{w}$ perpendicular to x_0:

$x_0 = \frac{x_1 + x_2}{2}$

$\vec{w} = x_2 - x_1$

for *$2 \times D$, $(l-1)$-level polytope* **do**

$y = (\mathcal{V} - x_0) * \vec{w}^T$

for $\sum_k y > 0$ **do**

for $\sum_j y < 0$ **do**

$\alpha = \frac{(x_0 - \mathcal{V}) * \vec{w}^T}{(\mathcal{V}_k^+ - \mathcal{V}_j^-) * \vec{w}^T}$

$u_i = \alpha * \mathcal{V}_k^+ + (1 - \alpha) * \mathcal{V}_j^-$

$m_i = \frac{1}{j+k} \sum_{j,k} u$

$\mathcal{D}_3^{\mathcal{P}} = Dist(m_i, u_i)$

$sort(\mathcal{D}_{\mathcal{P}}^3)$

Result: $child - \mathcal{P}$

are also calculated. The proposed algorithm evaluates the best quality for the current polytope Q. For all polytopes of a given decomposition level l, it calculates the highest ratio of the slope $S_{\mathcal{P}}$ at the mentioned polytopes positions:

$$Q = max(S_{b_1^{\mathcal{P}}}, S_{b_2^{\mathcal{P}}}, S_{c^{\mathcal{P}}}) \tag{11}$$

$$S_{b_1^{\mathcal{P}}} = \frac{f(\vec{b_1})}{\|\vec{b_1} - \mathcal{B}\|}, \quad S_{b_2^{\mathcal{P}}} = \frac{f(\vec{b_2})}{\|\vec{b_2} - \mathcal{B}\|}, \quad S_{c^{\mathcal{P}}} = \frac{f(c^{\mathcal{P}})}{\|c^{\mathcal{P}} - \mathcal{B}\|} \tag{12}$$

All child polytopes of a given level are sorted according to their quality and then saved to a list for later evaluation in the process. This list contains all child-polytopes of different levels $\{l-1, l-2, \cdots, 0\}$ are entirely visited until satisfying the stopping criterion or exploring the whole search space. Algorithm 4 details this process.

4.3. Local Search (Exploitation strategy)

A simple and deterministic heuristic local search (LS) is used to search the optimum in the child-polytopes of the last level. It is similar to the well-known Hooke-Jeeves Pattern Search method. This heuristic considers two solutions $\vec{x}_{a_1}$ and $\vec{x}_{a_2}$ for each dimension for evaluation, which are situated equidistant in reverse directions from the current solution x_a^{curr}, along any axis within the search space with a step size w:

$$\begin{aligned} \vec{x}_{a_1} &= x_a^{curr} + \omega \times \vec{e}_i \\ \vec{x}_{a_2} &= x_a^{curr} - \omega \times \vec{e}_i \end{aligned} \tag{13}$$

where $\vec{e}_i$ stands for the unit vector. The step size ω is adjusted to: (i) $\omega_{new} = \frac{\omega}{2}$, if any better alternative solution detected nearby $x_{v_i}^{cur}$, (ii) $\omega_{new} = \omega - \epsilon$, the step size value is gradually reduced until reaching a tolerance value ω_{min} (see Algorithm 3). The next current solution is then chosen as the optimal solution among $\vec{x}_{a_1}$, $\vec{x}_{a_2}$ and x_a^{curr}. Figure 4 depicts a flow chart diagram of polyFrac algorithm.

4.4. Convergence analysis of polyFrac

Here, the convergence property of the polyFrac algorithm is analyzed. We show that the probability P_ω that polyFrac converges from any point to any point in a regular or irregular polytope is not equal to zero. This property remains true even in a case where the promising point is placed at its extreme angle. Figure 3 illustrates an irregular polytope with 4 vertices limited with positive sides of 4 planes $y_1^+ = \vec{w}_1^T * (\mathcal{V}_1 - x)$, $y_2^+ =$

Algorithm 3: Local Search (LS)

Input: D: dimension, $w_{min} = 1 \times e^{-20}$: tolerance threshold

$\phi = 0.5$: step-size reduction

$\vec{c}$: search space barycenter

$\mathcal{N}_{eval} = 1$: number of the objective function evaluations

$f(\vec{c})$: initializing the latest best fitness value for corresponding position $\vec{c}$

current polytope $\mathcal{P}$ center $\vec{c} \leftarrow \vec{x}_c$

Evaluate the fitness of $\vec{x}_c$

Set the step size w to the longest distance between the farthest vertex v^{max} and the polytope center

while $\omega \geq \omega_{min}$ **do**
  *__for__ $i = 1, \cdots, D$ **do**
    $\vec{x}_R = \vec{x}_c + \omega \times \vec{e}_i$
    $\vec{x}_L = \vec{x}_c - \omega \times \vec{e}_i$
    $Sol^{best} \leftarrow$ best solution among $\{f(\vec{x}_R), f(\vec{x}_L), f(\vec{x}_c)\}$
    $\mathcal{N}_{eval} = \mathcal{N}_{eval} + 2$
    $\vec{x}_c \leftarrow Sol^{best}$
if *in level-l* $\vec{x}_c \in y^+$ **then**
  if $f(\vec{x}_c) == Sol^{best}$ **then**
    Reduce the step size $\omega = \omega \times \phi$
  if $f(\vec{x}_c) \leq Sol^{best}$ **then**
    Update the best solution $f(\vec{x}_c)$ and the best Position $\vec{x}_c$
else
  back to step * at the current level l

Result: $\vec{x}_c$

Algorithm 4: Move-up

Input: D dimension, l Current level

$N^{\mathcal{P}}_{l-1}$ number of explored polytopes at level $l-1$
$F_{max} = 5000 \times D$: stopping criterion

while $N == 2 \times D$ **do**
  $l = l$ - 1
  $N^{\mathcal{P}}_{l-1} \leftarrow N^{\mathcal{P}}_{l}$
if $l == 1$ & $\neq F_{max}$ **then**
  exploration of all polytopes
else
  at the current level l
  current polytope position ← next unexplored polytope

$\vec{w}_2^T * (\mathcal{V}_2 - x)$, $y_3^+ = \vec{w}_3^T * (\mathcal{V}_3 - x)$ and $y_4^+ = \vec{w}_4^T * (\mathcal{V}_4 - x)$. There are many paths connecting polytope centroid C to goal point $\mathcal{G}$. Let consider $Dist_{Manhattan}(C, \mathcal{G})$ the shortest path linking these two points which is calculated using the Manhattan distance. The probability of this path is denoted as P_ω:

$$P_\omega = (\frac{1}{D})^{N_\omega} \quad \text{where} \quad N_\omega = \frac{Dist_{\mathcal{M}}(C, \mathcal{G})}{\omega} \tag{14}$$

where D is the dimension, ω stands for the step size, and N_ω is the fraction of the Manhattan distance $\mathcal{M}$ between the centroid and the goal point. The probability of all possible paths between these points is represented by P_{all} which is the sum of P_ω and the probability of all remaining paths $P_{remains}$:

$$P_{all} = P_\omega + P_{remains} \tag{15}$$

Since $P_{remains} > 0$ is a strict positive value, thus $P_{all} > P_\omega$, then P_ω is a lower bound of P_{all}. This property is ensured using a condition (if $\vec{x}_c \in y^+$then) in Algorithm 3.

The probability of reaching $P^i(C, \mathcal{G})$ within i iterations is defined as:

$$P^i(C, \mathcal{G}) = \mathbb{E}\,(i \times P_\omega) \tag{16}$$

This property is also true for a regular polytope. Thus, polyFrac covers any point inside any polytope shape.

4.5. Sensitivity and Complexity analysis of polyFrac

A substantial step before fine-tuning the parameters of the algorithm, calibration is the sensitivity analysis of parameters.

To do so, the depth (k) is varied to measure the impacts of its fluctuations on polyFrac performance while keeping fix all other parameters with the values given in the previous section. The results are reported in Table 5. As these results show, sensitivity plays an important role in evaluating the performance of polyFrac. Indeed, the performance of the algorithm varies in terms of variation of the sensitivity. Consequently, k influences polyFrac performance and its suited value is 5 (this value is used in our experiments).

The asymptotic complexities of different parts of polyFrac are : fractal decomposition process $O(log_k(D))$, quality evaluation of a polytope (1), and local search application $O(log_2(r/\omega_{min}))$. Thus, the complexity of polyFrac is logarithmic, which depends on the fractal depth: $O(log_k(D) + 1 + log_2(\mathcal{D}/\omega_{min}))$ and a memory complexity of $\theta(D)$ where D stands for the dimension and $\mathcal{D}$ the Euclidean distance.

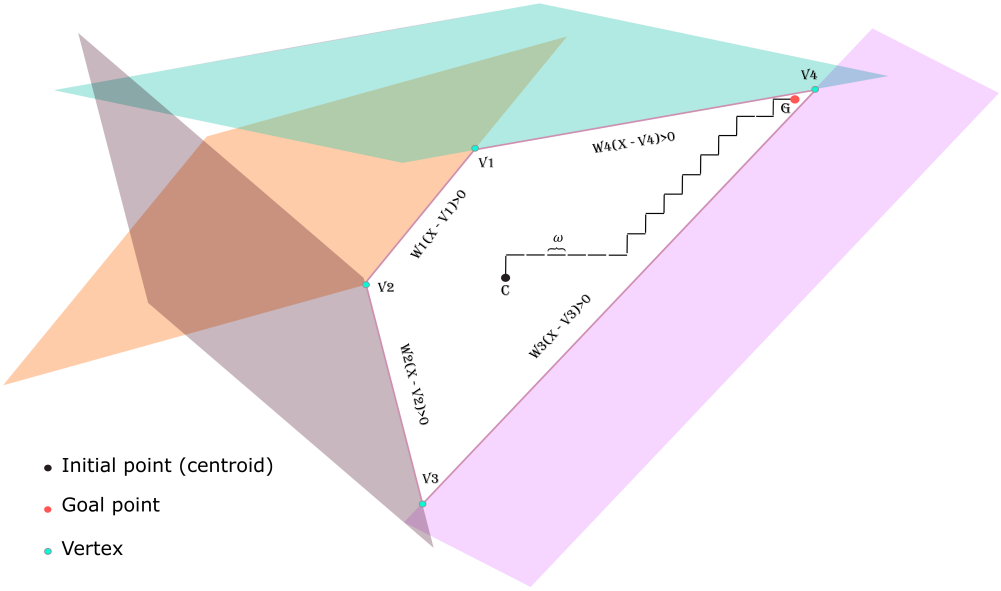

Figure 3: Example of an irregular polytope with 4 vertices limited with 4 planes $\vec{w}_1^T * (\mathcal{V}_1 - x)$, $\vec{w}_2^T * (\mathcal{V}_2 - x)$, $\vec{w}_3^T * (\mathcal{V}_3 - x)$ and $\vec{w}_4^T * (\mathcal{V}_4 - x)$. The step size is represented by ω.

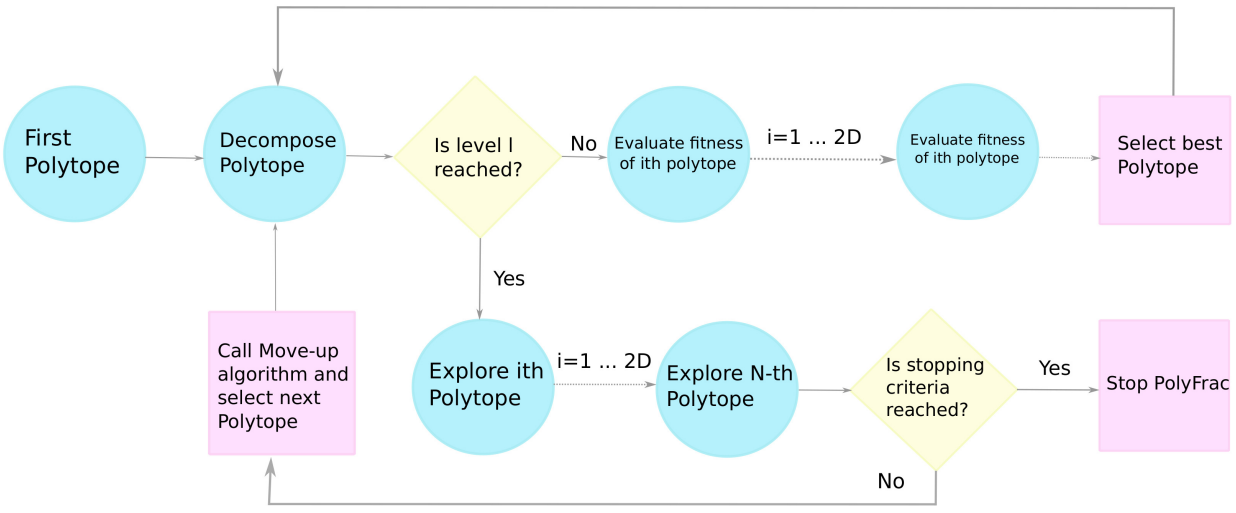

Figure 4: Flow chart diagram of polyFrac algorithm.

5. Results

The proposed algorithm, polyFrac, is evaluated on the well-known benchmarks to assess its performance. The following section briefly describes the benchmarks:

5.1. Test functions and parameters tuning

For the experimental study 19 functions (F_1–F_{19}) are selected for large scale continuous optimization problems from SOCO 2011 [29] which has many common functions with CEC 2013. The SOCO 2011 large-scale optimization benchmark has the advantage of covering different real-world scenarios where the substantial complexities of large-scale optimization problems are multi-modality and separability. It has been commonly used in various studies to assess the large-scale optimization algorithms performance [11]. The proposed algorithm performance is also tested on some selected benchmark functions from CEC 2017 [30]. These functions have a wide range of properties, such as disconnective, non-separable, partially separable, multimodal, degenerative, etc.

The definition of functions F_1–F_{11}, their features, and their properties are sketched in Table 1, Table 2, and Table 3, respectively. The non-separable functions F_{ns} (F_3,F_5,F_9,F_{10}) can be optimized dimension by dimension. The hybrid functions F_{12}–F_{19} are generated ($F_{ns} \oplus F'$) combining a non-separable function F_{ns} with a function among F_1,F_4,F_7, denoted as F'. This is done by splitting the solution the parameter m_{ns} into two parts; evaluating each of the parts using a function among F_{ns} and finally combining these results. The test function is assessed for dimensions: D = 50, 100, 200, 500 and 1000. The maximum number of function evaluations (FEs) is determined as the stopping criterion set to 5000 $\times D$ (from SOCO 2011 benchmark [29]). For each test function, the stochastic based algorithms are run 25 times. All parameters of polyFrac are empirically fitted as summarized in Table 4. It should be noted that the stopping criterion value for the LS is problem-dependent, and it is set to the shift values precision in the functions F_1–F_6. The step-size reduction value ϕ is set to 0.5.

5.2. PolyFrac results

The results of polyFrac are reported in Table 6 and Table 7 which are error values calculated using $f(x) - f(x^*)$ for dimensions 50D, 100D, 200D, 500D, and 1000D. As proposed in the benchmark functions paper [31], all average errors less than $1.0000E-14$ are seen as $0.0000E+00$. The standard deviations yield for all test functions are equal to $0.0000E+00$ which indicates that polyFrac regularly reaches the same optimum. The algorithm behavior for all functions is fully delineated by the mean and the standard deviation.

Overall, polyFrac shows good performance to reach the global optimum for the majority of test functions. However, it fails to solve Shifted Rosenbrock's function (F_3) and the hybrid functions implying F_3, F_{13}, and F_{17} (for dimension greater than 100D). This problem stems from the fact that algorithm LS (Algorithm(3)) is more appropriate for separable and semi separable problems due to its intrinsic nature.

5.3. Behavior analysis of polyFrac

This section analyses the polyFrac algorithm behavior in terms of exploration, exploitation operators, function evaluation consumption, fitness convergence, and the number of polytopes visited. The polyFrac algorithm behavior is analyzed regarding different function types: separable, semi separable, and non-separable functions. To this aim, three functions from each mentioned type is chosen from the benchmarks SOCO 2011 and CEC 2017 to investigate the proposed algorithm behavior. The lowest and the highest dimensions (50D and 1000D) were considered. The selected functions from benchmark SOCO 2011 are F_4 (separable), F_3 (nonseparable) and F_{16} (semi separable). The results are reported in Table 8 and Figure 5. Although non-separable function F_3 approaches the global optimum, it fails to reach it in all dimensions for the number of evaluation permitted in the experiments (250,000 evaluations for 50D and 5,000,000 evaluations for 1000D). The separable function F_4 reaches the global optimum after 5,001 evaluations for dimension 50D representing 2% of the stopping criterion. For 1000D, it reaches the optimum after 120,002 evaluations representing 2.4% of the permitted function evaluations. The semi separable function F_{16} reaches the optimum after 25,200 function evaluations for dimension 50D representing 10.08% of the allowed evaluation and 280,001 function evaluations for 1000D representing 5.6% of the stopping criterion. These results show the stability and scalability of polyFrac. In the exploration phase, the number of visited polytopes is the same in each function for both dimensions (50D and 1000D). This fact reveals the stability and scalability of the polyFrac algorithm, regardless of the dimension. Table 8 and Figure 5 also illustrate the number of polytopes visited at all l levels.

The fitness convergence of polyFrac is analyzed on CEC 2017 benchmark functions. Figure 8 and Figure 9 depict the mean error over the number of function evaluations. Table 17 reports a comparison between polyFrac and most efficient PSO-based algorithms over

Table 1: Functions F_1-F_{11}.

Function	Name	Expression
F_1	Shifted sphere function	$\sum_{i=1}^{D} z_i^2 + f_{\text{bias}}, z = x - o$
F_2	Shifted Schwefel Problem 2.21	$max\{\|z_i\|, 1 \le i \le D\} + f_{\text{bias}}, z = x - o$
F_3	Shifted Rosenbrock's Function	$\sum_{i=1}^{D-1}(100(z_i^2 - z_{i+1})^2 + (z_i - 1^2)) + f_{\text{bias}}, z = x - o$
F_4	Shifted Rastrigin's Function	$\sum_{i=1}^{D}(z_i 2 - 10cos(2\pi z_i) + 10) + f_{\text{bias}}, z = x - o$
F_5	Shifted Griewank's Function	$\sum_{i=1}^{D} \frac{z_i^2}{4000} - \prod_{i=1}^{D} cos(\frac{z_i}{\sqrt{i}}) + 1 + f_{\text{bias}}, z = x - o$
F_6	Shifted Ackley's Function	$-20.exp(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} z_i^2} - exp(\frac{1}{D}\sum_{i=1}^{D} cos(2\pi z_i)) + 20 + \mathbf{e} + f_{\text{bias}}, z = x - o$
F_7	Schwefel's Problem 2.22	$\sum_{i=1}^{D} \|x_i\| + \prod_{i=1}^{D} \|x_i\|$
F_8	Schwefel's Problem 1.2	$\sum_{i=1}^{D}(\sum_{j=1}^{D} x_j)$
F_9	Extended f10	$\sum_{i=1}^{D-1} f_{10}(x_i, x_{i+1})) + f_{10}(x_D, x_1)$ $f_{10}(x, y) = (x^2 + y^2)^{0.25}(sin^2(50(x^2 + y^2)^{0.1}) + 1)$
F_{10}	Bohachevsky	$\sum_{i=1}^{D-1}(x_i^2 + 2x_{i+1}^2 - 0.3cos(3\pi x_i) - 0.4cos(4\pi x_{i+1}) + 0.7)$
F_{11}	Schaffer	$(x_i^2 + x_{i+1}^2)^{0.25}(sin^2(50(x_i^2 + x_{i+1}^2)^{0.1}) + 1)$

Table 2: Properties of functions $F_1 - F_{15}$.

Function	Range	Optimum $f(x^*)$	Unimodal/Multimodal	Shifted	Separable	Optimizable dimension by dimension
F_1	$[-100, 100]^D$	-450	U	✓	✓	✓
F_2	$[-100, 100]^D$	-450	U	✓		
F_3	$[-100, 100]^D$	390	M	✓		✓
F_4	$[-5, 5]^D$	-330	M	✓	✓	✓
F_5	$[-600, 600]^D$	180	M	✓		
F_6	$[-32, 32]^D$	-140	M	✓	✓	✓
F_7	$[-10, 10]^D$	0	U		✓	✓
F_8	$[-65.536, 65.536]^D$	0	U			
F_9	$[-100, 100]^D$	0	U			✓
F_{10}	$[-15, 15]^D$	0	U			
F_{11}	$[-100, 100]^D$	0	U			

these functions where the polyFrac algorithm behavior is similar across dimensions. In the exploitation phase, the local search is used to seek the best solution (or for the global optimum). The slope descends steeply to reach the best solution or the optimal solution for F_6. The sudden change in the curves occurs when the local search algorithm starts, as in the case of multi-modal functions (*e.g.* F_9). As shown in Figure 8, after an abrupt tumble, the curve stagnated until the stopping criterion is reached. In summary, polyFrac shows a stable and scalable behavior across all dimensions in the case of separable, weakly separable, uni-modal, and multi-modal functions, keeping the number of visited polytopes constant and the percentage of allowed function evaluations constant as well.

5.4. Comparison with competing algorithms

The polyFrac performance is compared with a deterministic metaheuristic for large-scale Optimization (FDA) [11], and DIRECT [7] as related algorithms. It is compared to other competing optimization algorithms from the literature for SOCO 2011 benchmarks.

5.4.1. Comparison of polyFrac with DIviding RECTangles (DIRECT)

DIRECT is a well-known multi-scale optimization algorithm that decomposes the search space to find the global optimum. The comparison is conducted for functions F_1 to F_6 and for dimensions $50D$ and $100D$. This choice is due to the fact that the number of expansion increases in a quadratic way regarding the problem dimension. As shown in Table 9, the DIRECT algorithm fails to reach the global optimum for dimension $D > 10$ and polyFrac outperforms DIRECT for all benchmark functions and all dimensions.

Table 3: Properties of functions $F_{12} - F_{19}$.

Function	F_{ns}	**F'**	m_{ns}	**Range**	**Optimum** $f(x^*)$
F_{12}	$NS^* - F_9$	F_1	0.25	$[-100, 100]^D$	0
F_{13}	$NS - F_9$	F_3	0.25	$[-100, 100]^D$	0
F_{14}	$NS - F_9$	F_4	0.25	$[-5, 5]^D$	0
F_{15}	$NS - F_{10}$	$NS - F_7$	0.25	$[-10, 10]^D$	0
F_{16}	$NS - F_9$	F_1	0.75	$[-100, 100]^D$	0
F_{17}	$NS - F_9$	F_3	0.75	$[-100, 100]^D$	0
F_{18}	$NS - F_9$	F_4	0.75	$[-5, 5]^D$	0
F_{19}	$NS - F_{10}$	$NS - F_7$	0.75	$[-10, 10]^D$	0

* non-separable

Table 4: Parameters tuning of polyFrac.

Parameters	**Value**	**Description**
k	5	Decomposition level
ω_{min}	$1 \times \mathbf{e}^{-20}$	Stopping criterion for LS
ϕ	0.5	step-size reduction

5.4.2. Comparison of polyFrac with Fractal Decomposition Algorithm (FDA)

As discussed in sub-section 3.3, FDA is an algorithm decomposing the search space into hyperspheres. For the sake of brevity, the comparison is conducted for the lowest and the highest dimensions $50D$ and $1000D$ and for functions F_2, F_3, F_6, F_{13} and F_{17} for which FDA did not reach the global optimum. As reported in Table 10, the polyFrac algorithm outperforms FDA in all functions and dimensions. Indeed, despite inflated hyperspheres, the FDA fails to fully cover the search space. Besides, due to this inflation, it can search iteratively in overlapping regions or out of space and fails to find the global optimum before reaching the stop criteria.

5.4.3. Comparison of PolyFrac with SOCO 2011 contributors

In this section, the polyFrac algorithm is compared with SOCO 2011 contributors. Hybrid metaheuristics are excluded from this study since they belong to a specific metaheuristic category. The following algorithms are selected:

- Differential Evolution Algorithm (DE) [32] using the exponential crossover.
- Real-coded Genetic Algorithm (CHC) [33]
- Memetic algorithm based on local search chains (MA-SSW-Chains) [34] using memetic algorithm with a local search.
- Multiple Trajectory Search for large-scale optimization (MTS-LS1) [35] using the appropriate values suggested in [36].
- Self-adaptive Differential Evolution (SaDE) [37] using a learning phase to generate vectors strategies.
- Multi-population Differential Evolution with balanced ensemble of mutation strategies for large-scale optimization (mDE-bES) [38]. This algorithm uses various mutation operators for each divided space and updates the strategies during the search.
- Self-adaptive differential evolution algorithm (jDElscop) using three strategies, and a mechanism to decrease the population size [39].

The complexities of the algorithms are reported in Table 11. According to this table, the lowest complexity belongs to polyFrac and FDA. While these latter algorithms have logarithmic complexities, the other ones have polynomial complexities.

The algorithms' performances for all dimensions are compared using Friedman rank-sum and the results are presented in Table 12 and Figure 6. The results show that polyFrac is ranked first within all dimensions. Indeed, for each benchmark function, the average relative rank is calculated based on the average performance of each algorithm. An average rank is reported through all the functions.

Table 5: Analysis of the Sensitivity with regards to the fractal depth (k). The average error is reported for 200D, 500D and 1000D.

	200D				500D				1000D			
k	3	4	5	6	3	4	5	6	3	4	5	6
F_2	3.29E-12	**2.12E-14**	3.09E-14	2.98E-11	3.83E-10	2.76E-11	**4.09E-12**	1.88E-06	2.97E-07	2.12E-07	8.23E-08	**7.43E-08**
F_3	2.33E-02	4.31E-04	**9.05E-07**	6.77E-03	4.23E-01	1.61E-02	**7.33E-05**	2.98E-03	1.44E-04	2.43E-01	**6.21E-04**	3.67E-02
F_6	2.21E-13	8.93E-14	**1.88E-14**	5.01E-07	**3.01E-14**	4.23E-14	2.74E-13	2.89E-10	4.13E-13	**3.71E-14**	9.11E-13	7.82E-10
F_{13}	5.21E-04	3.35E-04	**5.11E-05**	8.21E-04	4.33E-03	1.93E-02	9.01E-04	**1.22E-04**	8.21E-04	5.01E-03	**8.19E-05**	4.13E-04
F_{17}	1.28E-01	**3.16E-06**	4.12E-02	1.81E-01	7.02E-01	**4.31E-05**	4.81E-03	6.19E-02	3.47E-01	**1.00E-04**	5.54E-01	3.37E-02

Values in bold represent the best result for a given function and dimension across all different values of k.

Table 6: Results achieve by polyFrac on functions F_1-F_{10}.

Dimension	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}
50D	0.00E+00	1.33E-14	5.90E-01	0.00E+00	0.00E+00	5.01E-14	0.00E+00	0.00E+00	0.00E+00	0.00E+00
100D	0.00E+00	4.30E-14	6.09E-02	0.00E+00	0.00E+00	2.06E-14	0.00E+00	0.00E+00	0.00E+00	0.00E+00
200D	0.00E+00	1.02E-11	0.00E+00	0.00E+00	0.00E+00	1.62E-13	0.00E+00	0.00E+00	0.00E+00	0.00E+00
500D	0.00E+00	9.00E-10	0.00E+00	0.00E+00	0.00E+00	7.27E-13	0.00E+00	0.00E+00	0.00E+00	0.00E+00
1000D	0.00E+00	9.46E-05	0.00E+00	0.00E+00	0.00E+00	9.27E-13	0.00E+00	0.00E+00	0.00E+00	0.00E+00

The raw and adjusted ρ-values of the Wilcoxon test are presented in Table 13 and Table 14. Therefore, algorithms with a p-value < 0.05 are statistically outperformed by the proposed algorithm. As shown in these tables, polyFrac outperforms FDA, MA-SSW-Chains, CHC, mDE-bES, DE, MTS-LS1, and SaDE in all dimensions (algorithms with a ρ-value < 0.05). As shown in Table 15, polyFrac reaches approximately 15 times the global optimum for all the benchmarks dimensions. While it solves 15 problems out of 19, the second and third highest performances are obtained for mdE-bES and jDElscop where they solve only 9 problems. All these results show that polyFrac yields good performance for the benchmark functions in terms of scalability and is stable across all dimensions. Besides, the analysis of polyFrac demonstrates that it is an effective approach to solving problems on a large scale. Table 18 reports the frequency with which these algorithms have reached the global optimum for all the benchmarks functions.

The tables 20-25 compare the average error obtained by applying polyFrac and the algorithms mentioned on the reference functions for all dimensions. Fig 10 portrays the average rank distributions for each algorithm on all functions. As shown in these boxplots, polyFrac has a more stable performance than the other algorithms for all functions within all dimensions. A Wilcoxon pairwise test is conducted on polyFrac and all algorithms for a qualified comparison of the algorithms' performances. The Holm procedure [40] for the familywise error rate is used to adjust the ρ-values provided by the Wilcoxon test.

5.4.4. *Comparison of polyFrac with metaheuristics methods*

For a comparison of the proposed approach with large-scale global metaheuristics, the following algorithms are selected from [40]: Deterministic metaheuristic based on Fractal Decomposition for large-scale Optimization (FDA) [11], Large Scale Global Optimization with MOS-based hybrid algorithms (MOS-CEC2013), Multiple Offspring Sampling in Large Scale Global Optimization (MSO-CEC2012), Multiple Offspring Sampling (MSO-SOCO2011), IACO R-Hybrid, Two-stage based ensemble optimization (X2S.Ensemble). Table 25 presents the average errors of the aforementioned algorithms using 19 benchmark functions for dimension 50D.

The Friedman rank-sum test is shown in Table 19 (first column) where polyFrac is ranked first. The ρ-values is reported in Table 19 (second column). This value is obtained using a Wilcoxon pairwise test and is adjusted using the Holm procedure. As shown in this Table 19 and in Figures 7-10, the polyFrac algorithm more efficient than FDA, MOS-CEC2013, MOS-CEC2012, MOS-SOCO2011, and 2S-Ensemble. The adjusted ρ-values obtained from the Wilcoxon test confirms polyFrac efficiency. Furthermore, it shows stable performance for 15 functions, which is more than all other algorithms (see Table 18).

5.4.5. *Comparison of polyFrac with recent metaheuristics methods*

The polyFrac algorithm performance is compared with recent algorithms applied on CEC 2017 benchmark functions [30] to make a comprehensive comparison.

These algorithms are briefly described below:

Table 7: Results achieved by polyFrac on functions F_{11}-F_{19}.

Dimension	F_{11}	F_{12}	F_{13}	F_{14}	F_{15}	F_{16}	F_{17}	F_{18}	F_{19}
50D	0.00E+00	0.00E+00	7.81E-14	0.00E+00	0.00E+00	0.00E+00	5.90E-01	0.00E+00	0.00E+00
100D	0.00E+00	0.00E+00	1.92E-13	0.00E+00	0.00E+00	0.00E+00	6.09E-02	0.00E+00	0.00E+00
200D	0.00E+00	0.00E+00	3.59E-13	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
500D	0.00E+00	0.00E+00	9.02E-13	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
1000D	0.00E+00	0.00E+00	1.97E-12	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00

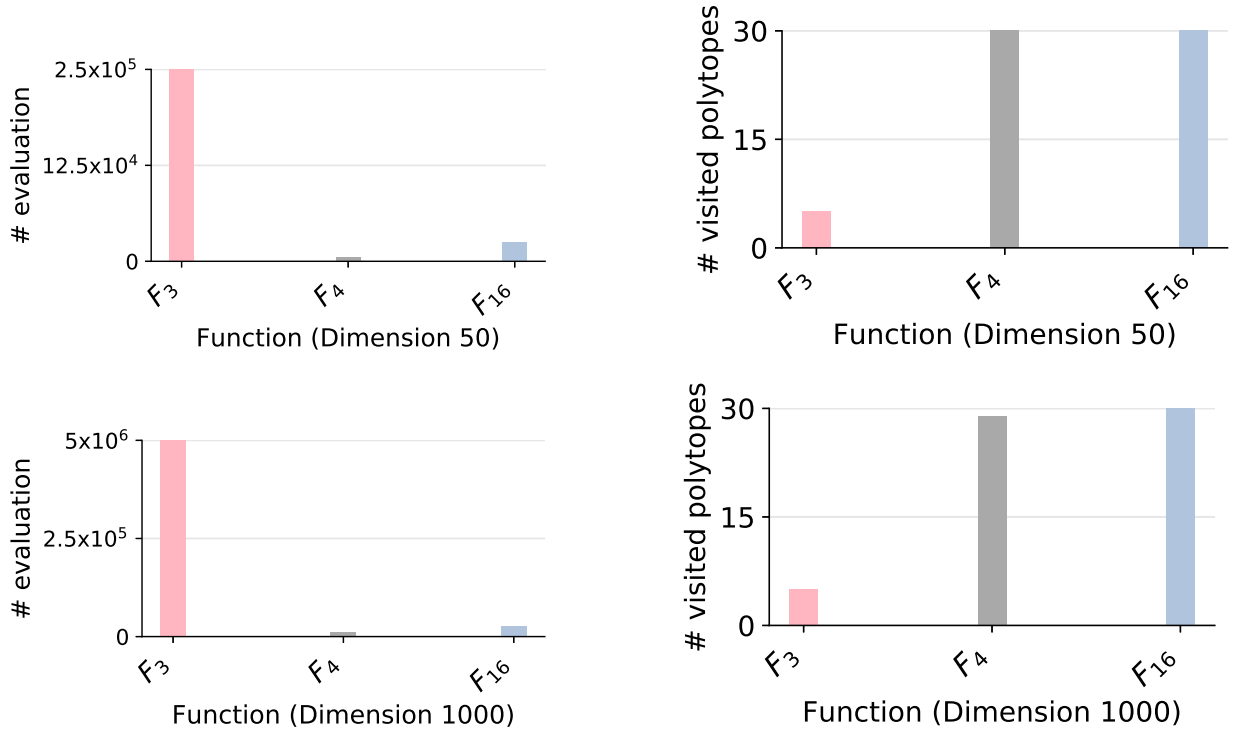

Figure 5: Number of evaluations (left) and visited polytopes (right) to find the best solution for functions F_3, F_4 and F_{16} for D = 50 and D = 1000.

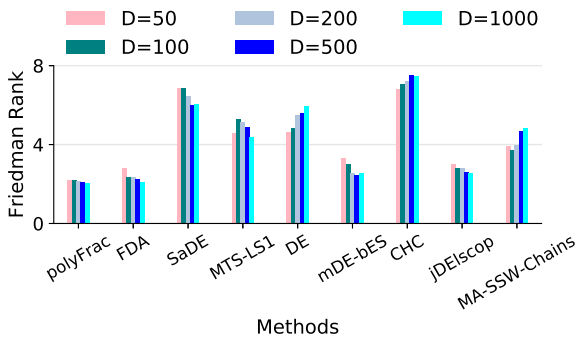

Figure 6: Comparative performance of the algorithms on Friedman rank for dimensions 50D, 100D, 200D, 500D and 1000D.

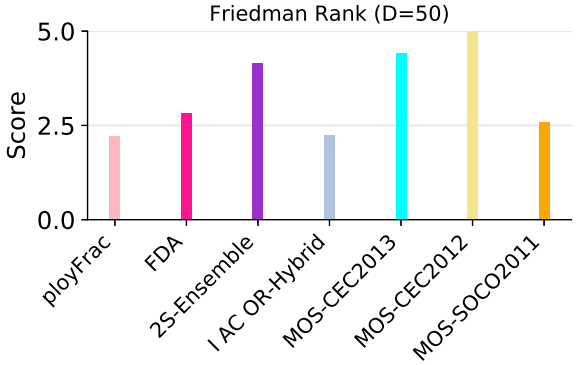

Figure 7: Comparative performance of the algorithms on Friedman rank for dimensions 50D.

Table 8: Number of evaluations and visited polytopes to find the best solution for F_3, F_4 and F_{16} for dimensions $50D$ and $1000D$.

	$50D$		$1000D$	
Function	№ Evaluation	№ Visited $\mathcal{P}$	№ Evaluation	№ Visited $\mathcal{P}$
F_3	250,000	5	5,000,000	5
F_4	5,001	30	120,002	29
F_{16}	25,200	33	280,001	33

- LSHADE-SPACMA (LSSPA) [41]: mixes LSHADE-SPA and CMA-ES algorithms.
- jSO: [42]: ameliorates variant of iL-SHADE algorithm.
- MM-OED (MM) [43]: is based on a multi-method algorithm using orthogonal experiment design (OED).
- MOS (MOS11, MOS12, MOS13) [44]: are three algorithms for large-scale global optimization.
- PPSO [18]: relies on Particle Swarm Optimization using fuzzy logic.
- DYYPO [45]: modifies Yin-Yang Pair Optimization algorithm by integrating dynamic archive updating strategy.
- RB-IPOP-CMA-ES (RBI) [46]: modifies IPOP-CMA-ES algorithm by restarting the trigger in accordance with the fitness.
- TLBO-FL (TFL) [47]: modifies the Teaching Learning Based Optimization algorithm by including focused learning of students.

The test functions of CEC 2017 are categorized into four classes as uni-modal functions (F_1-F_3), multi-modal functions (F_4-F_{10}), hybrid functions (F_{11}-F_{20}) and composition functions (F_{21}-F_{29}) [45]. From each class, three functions are selected to evaluate the performance of polyFrac for dimension $100D$. A comparison of the mean error values obtained for $100D$ between polyFrac and all mentioned algorithms are given in Figures 8 and 9. The polyFrac algorithm is ranked first among all test functions (this information is obtained from [48]).

As shown in this figure, polyFrac converges faster than the other algorithms and has the smallest mean values for most of the functions.

5.4.6. *Comparison of polyFrac with PSO-based methods*

The polyFrac algorithm performance is compared with the following swarm intelligence algorithms applied on CEC 2017 benchmark functions [30]. These algorithms are briefly described below:

- HSSAPSO [19]: combines the salp swarm algorithm with PSO to remedy their drawbacks and take advantage of the two algorithms.
- MPSO [16]: improves local search ability of the standard PSO algorithms.
- HFPSO [17]: determines the start of the local search process by checking the previous global best fitness values.
- PPSO [18] relies on PSO using fuzzy logic.

A comparison of the mean error values obtained for $100D$ between polyFrac and all mentioned algorithms are given in Table 17. The polyFrac algorithm outperforms almost all the PSO-based algorithms on the test functions. The results of all competing algorithms are taken from their original papers. One can see that polyFrac outperforms significantly all PSO-based methods in most of all test functions.

5.4.7. *Applying polyFrac on real-world optimization problem*

In this section, a real-world optimization problem is used to test the performance of the proposed algorithm. The selected problem concerns the future traffic speed prediction of the road segments (links) based on the traffic flow data [49, 50]. The deep learning architecture proposed in [49, 50] is a convolutional attention-based gated recurrent unit (GRU) minimizing Mean Square Error (MSE), defined as follows:

$$\mathcal{M} = \frac{\sum_{i=1}^{N} [s_i^{(n)} - \hat{s}_i^{(n)}]^2}{\mathcal{N}} \tag{17}$$

where $\hat{s}_i^{(n)}$ is the estimated speed for the input traffic flow $f_i^{(n)}$, $s_i^{(n)}$ is the real traffic speed, *i.e.* ground truth. The total number of samples is denoted with $\mathcal{N}$ which is 18,162 samples for an aggregation rate of 2 hours' traffic data. To minimize $\mathcal{M}$ stochastic gradient descent via a back-propagation process updates all learning parameters of the model Θ, *i.e.* weights in the opposite direction:

$$\nabla\Theta = -\gamma \frac{\delta \mathcal{L}}{\delta \Theta} \tag{18}$$

where γ stands for the learning rate determining the steps' size to reach a local minimum. In the proposed deep neural network's architecture 1952 weights parameters are used, which is calculated based on the network structure.

Table 9: Comparison of polyFrac and DIRECT algorithms for dimensions $50D$ and $100D$.

	$50D$		$100D$	
Function	DIRECT	polyFrac	DIRECT	polyFrac
F_1	5.53E+01	**0.00E+00**	1.06E+04	**0.00E+00**
F_2	5.53E+01	**1.33E-14**	7.45E+01	**4.30E-14**
F_3	1.79E+04	**5.90E-01**	9.84E+07	**6.09E-02**
F_4	1.26E+02	**0.00E+00**	6.64E+02	**0.00E+00**
F_5	1.06E+00	**0.00E+00**	5.06E+01	**0.00E+00**
F_6	1.59E-01	**5.01E-14**	1.87E-01	**2.06E-14**

Table 10: Comparison of polyFrac and FDA algorithms for dimensions $50D$ and $1000D$.

	$50D$		$1000D$	
Function	FDA	polyFrac	FDA	polyFrac
F_2	2.71E-12	**1.33E-14**	3.11E-01	**9.46E-05**
F_3	5.53E+01	**5.90E-1**	1.13E+03	**0.00E+00**
F_6	**5.75E-14**	**5.01E-14**	1.91E-12	**9.27E-13**
F_{13}	5.50E+01	**7.81E-14**	7.69E+02	**1.97E-12**
F_{17}	**6.09E-04**	5.90E-01	1.95E+02	**0.00E+00**

Table 11: Comparison of the complexity of polyFrac with algorithms in the literature.

Algorithm	**Complexity**
DE	$O(D^2)$
CHC	$O(D^2)$
MTS-LS1	Less than or equal to $O(D^2)$
SaDE	Less than or equal to $O(D^3)$
mDE-bES	Less than or equal to $O(D^2)$
jDElscop	Less than or equal to $O(D^2)$
MA-SSW-Chains	Greater than or equal to $O(D^3)$
FDA	Less than or equal to $O(log_k(D))$
polyFrac	Less than or equal to $O(log_k(D))$

More detail on the network's structure is given in [49, 50]. The polyFrac algorithm is used in this deep neural network to compute and update the weights, Θ, throughout different epochs. The framework converges after about 150 epochs in this experiment (each epoch takes almost 2 minutes). The performance of the polyFrac algorithm used in the neural network's structure is evaluated using the conventional metrics:
Mean Absolute Error (MAE):

$$MAE = \frac{\sum_i^n |y_i - \hat{y}_i|}{n}, \tag{19}$$

Root Mean Square Error (RMSE) :

$$RMSE = \sqrt{\frac{\Sigma_i (y_i - \hat{y}_i)^2}{n}}, \tag{20}$$

and Mean Absolute Percentage Error (MAPE):

$$MAPE = \sum_i^n |\frac{y_i - \hat{y}_i}{y_i}| \frac{100\%}{n}, \tag{21}$$

where n denotes the total number of prediction points, y_i stands for the ground truth values, and $\hat{y}_i$ denotes the prediction values. The performance is then compared with the initial neural network, *i.e.* without using polyFrac. The results are reported in Table 16. As shown, the network using polyFrac to update its weights outperforms the initial network.

6. Conclusion

In this paper, we proposed a new metaheuristic algorithm based on the fractal decomposition approach. Herein, we presented a new method that makes use of hyper-polytopes to decompose the search space. The main contribution is the polyFrac algorithm, proposing a novel hyper-polytope decomposition for continuous optimization. Indeed, this method recursively decomposes the search space to non-overlapping regions and it is possible to move throughout different granularity

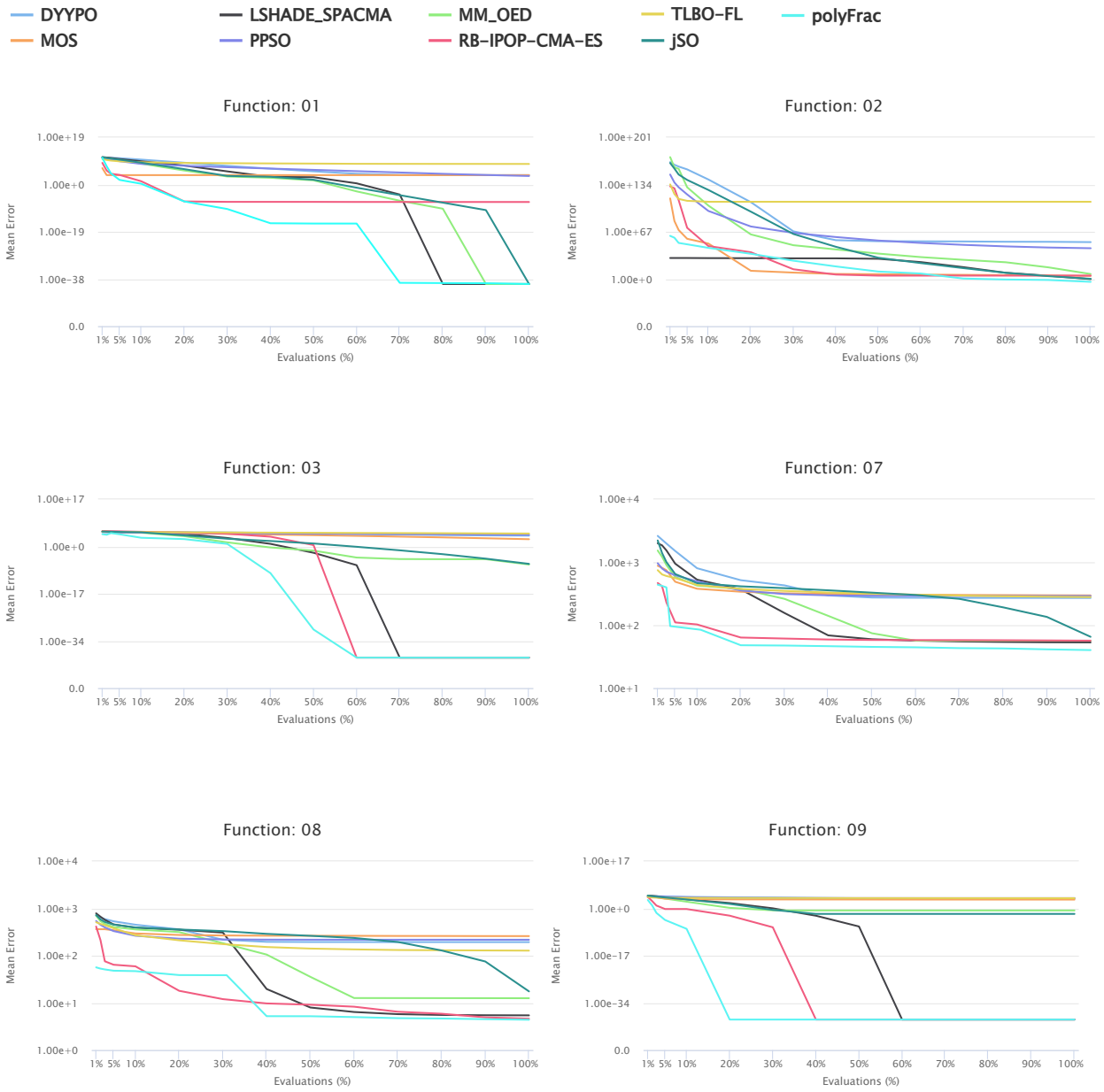

Figure 8: Mean error values for functions $F_1, F_2, F_3, F_7, F_8, F_9$ for all algorithms from CEC 2017 competition and polyFrac for dimension $100D$.

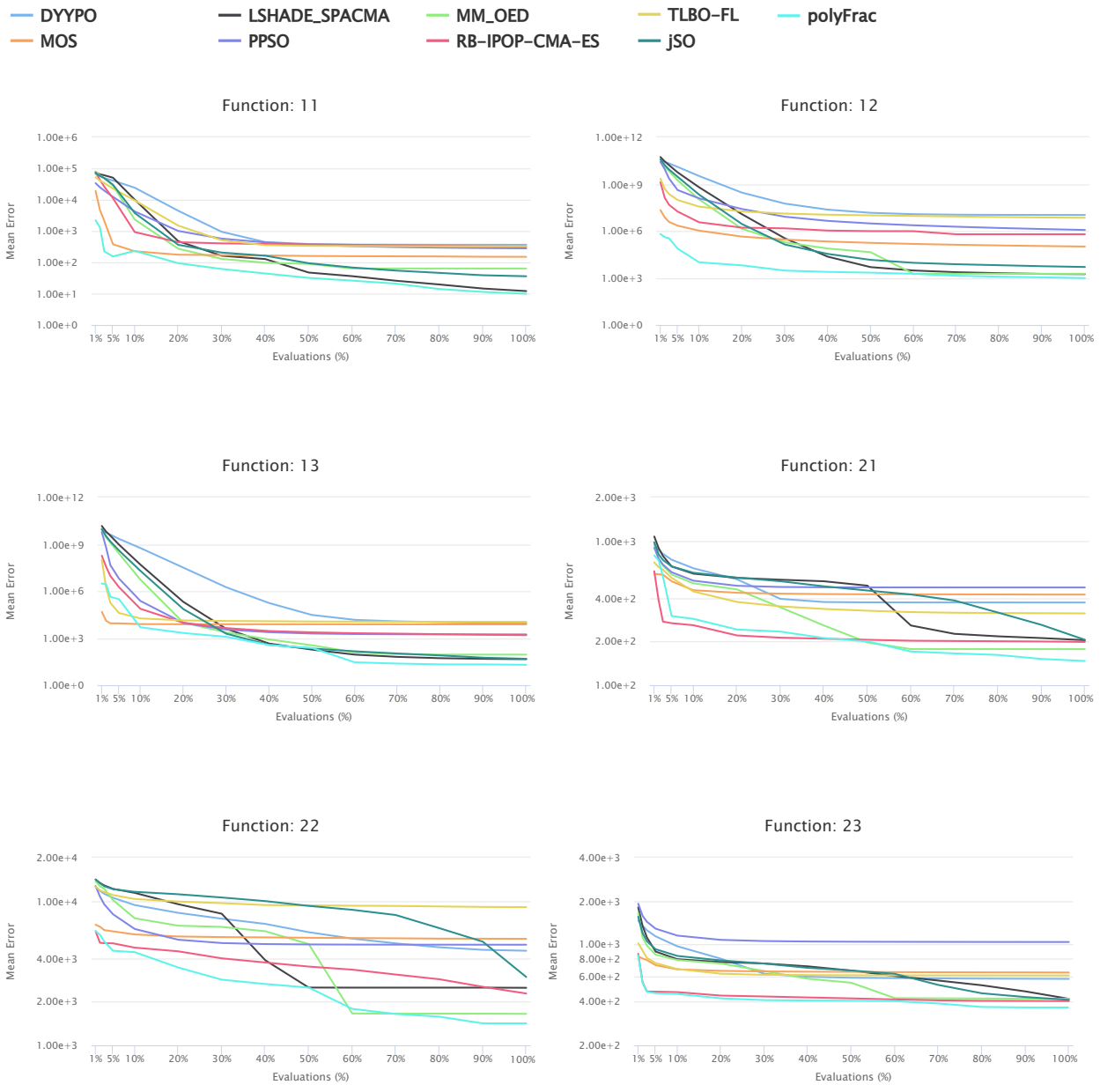

Figure 9: Mean error values for functions $F_{11}, F_{12}, F_{13}, F_{21}, F_{22}, F_{23}$ for all algorithms from CEC 2017 competition and polyFrac for dimension $100D$.

Table 12: Friedman Rank sum for all algorithms at dimensions 50D, 100D, 200D, 500D and 1000D.

Algorithm	50D	100D	200D	500D	1000D
MA-SSW-Chains	3.92	3.74	3.97	4.68	4.84
jDElscop	3.00	2.82	2.79	2.61	2.55
CHC	6.84	7.05	7.24	7.53	7.47
mDE-bES	3.32	3.03	2.55	2.47	2.58
DE	4.63	4.82	5.50	5.58	5.97
MTS-LS1	4.58	5.29	5.13	4.87	4.39
SaDE	6.89	6.89	6.45	6.00	6.08
FDA	2.82	2.37	2.37	2.26	2.11
polyFrac	**2.29**	**2.03**	**1.84**	**1.98**	**2.03**

Table 13: Raw and adjusted ρ-values from Wilcoxon test using the Holm procedure for dimensions 50D, 100D and 200D.

	50D		100D		200D	
polyFrac vs.	Raw	Adjusted	Raw	Adjusted	Raw	Adjusted
MA.SSW.Chains	2.21E-07	3.43E-07	1.11E-09	4.01E-08	5.87E-08	1.98E-08
jDElscop	**2.01E-01***	**2.01E-01***	**1.32E-01***	**1.86E-01***	**1.88E-01***	**2.71E-01***
CHC	2.32E-9	1.13E-9	2.53E-10	5.12E-10	2.11E-10	4.28E-10
mDE.bES	1.12E-04	3.42E-04	1.12E-03	1.12E-03	3.52E-02	3.52-02
DE	2.12E-07	1.63E-07	4.98E-09	1.15E-08	6.65E-09	2.32E-08
MTS.LS1	4.11E-08	1.45E-06	2.01E-08	1.03E-07	3.17E-08	1.02E-07
SaDE	3.23E-09	2.07E-09	1.13E-09	5.59E-09	2.78E-09	1.34E-08
FDA	2.34E-09	1.07E-08	3.03E-09	6.01E-09	2.90E-09	1.18E-08

* ρ-value > 0.05 fails to indicate the difference with significant level $\alpha = 0.05$

levels by only computing the average of vertices of a polytope to obtain the coordinates of the centroids. This property allows reducing the computation complexity of the algorithm when the size of the search space increases. Then, the most promising hyper-polytopes are iteratively decomposed into child-polytopes. Finally, a simple deterministic local search (metaheuristic based on a single solution) is used to intensively search for the best solution within the selected low-level hyper-polytope. The polyFrac backtracks via the moving-up procedure to explore the rest of the search space until the stopping criterion is satisfied to avoid being stall into a local optimum. The proposed algorithm was tested on different test functions from large scale continuous optimization problems. The obtained results show the efficiency of the proposed algorithm and the comparisons with other recent and state-of-the-art algorithms prove that its performance is very competitive for all considered dimensions. Then, we intend to introduce a parallel population-based method to enhance the algorithm performance dealing with dynamic optimization problems. In this study, we focused on a classical method to put forward the proposed algorithm sensitivity. Instead, a metaheuristic will be used to optimize hyperparameters in future work. Two other applications on real-world problems are also in development.

7. Statement

This manuscript is the authors' original work and has not been published nor has it been submitted simultaneously elsewhere. All authors have checked the manuscript and have agreed to the submission.

References

[1] J. Del Ser, E. Osaba, D. Molina, X.-S. Yang, S. Salcedo-Sanz, D. Camacho, S. Das, P. N. Suganthan, C. A. C. Coello, F. Herrera, Bio-inspired computation: Where we stand and what's next, Swarm and Evolutionary Computation 48 (2019) 220–250.

[2] I. Fister Jr, X.-S. Yang, I. Fister, J. Brest, D. Fister, A brief review of nature-inspired algorithms for optimization, arXiv preprint arXiv:1307.4186.

Table 14: Raw and adjusted ρ-values from Wilcoxon test using the Holm procedure for dimensions $500D$ and $1000D$.

	$500D$		$1000D$	
polyFrac vs.	Raw	Adjusted	Raw	Adjusted
MA.SSW.Chains	3.90E-08	1.15E-07	2.33E-09	3.19E-09
jDElscop	**2.44E-01***	**1.11E-01***	**4.54E-01***	**1.03E-01***
CHC	6.13E-10	4.98E-08	5.36E-10	3.17E-08
mDE.bES	5.33E-04	5.53E-04	1.88E-04	1.88-04
DE	2.53E-04	2.53E-04	1.77E-04	1.77E-04
MTS.LS1	2.03E-07	1.67E-05	2.23E-06	2.99E-05
SaDE	3.83E-09	1.34E-08	2.88E-09	1.32E-07
FDA	2.93E-10	1.12E-09	1.12E-10	1.34E-09

* ρ-value > 0.05 fails to indicate the difference with significant level $\alpha = 0.05$

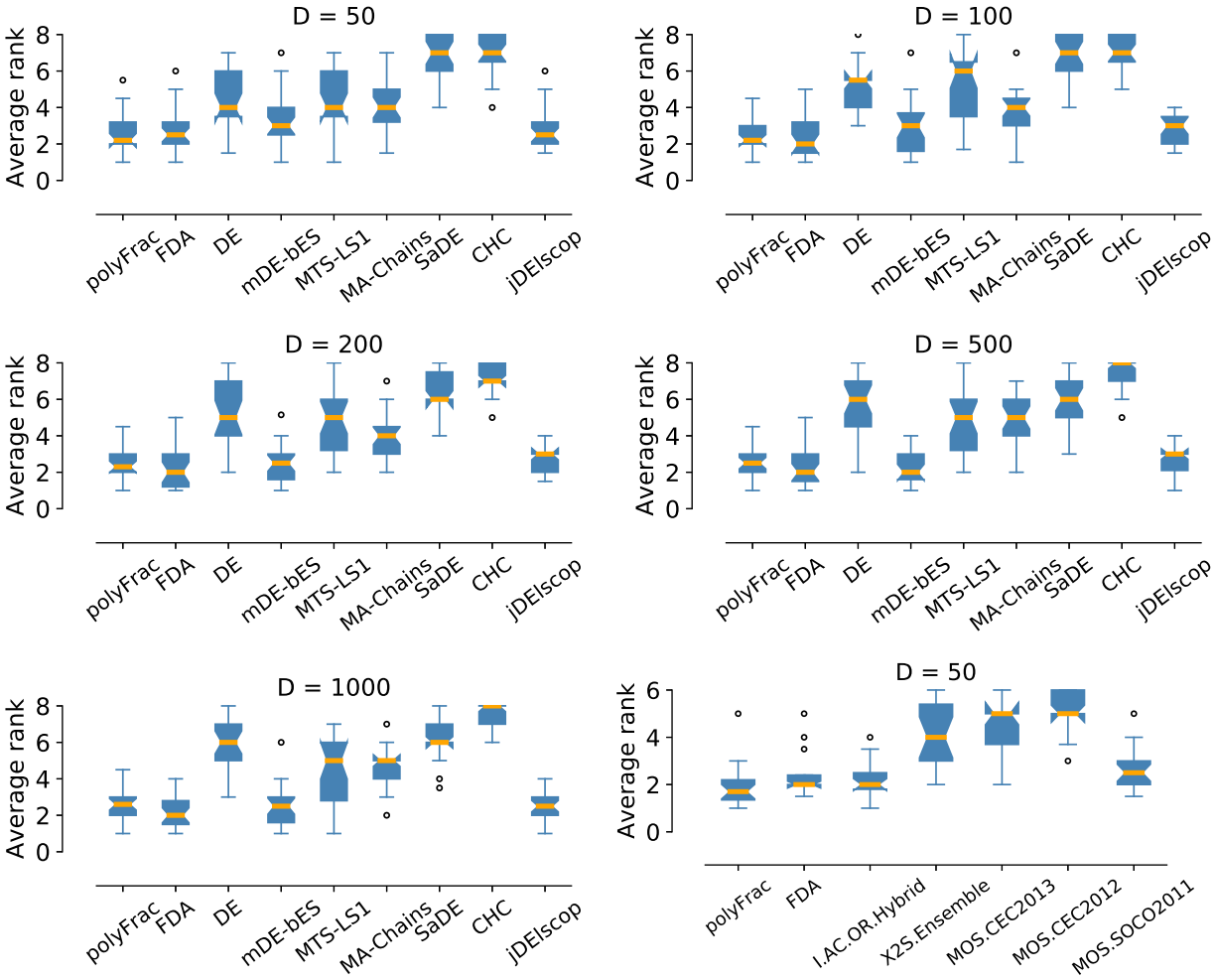

Figure 10: Comparison of the average rank distributions for all algorithms.

Table 15: The frequency at which the global optimum is reached.

Algorithm	50D	100D	200D	500D	1000D	**AVERAGE**	**RANK**
MA-SSWChains	9	8	6	3	2	5.6	(5)
jDElscop	12	10	9	8	7	9.2	(4)
CHC	0	0	0	0	0	0	(9)
mDE-bES	9	9	11	9	9	9.4	(3)
DE	7	4	2	1	1	3	(7)
MTS-LS1	7	4	4	2	4	4.2	(6)
SaDE	1	1	1	1	1	1	(8)
FDA	14	14	14	14	14	14	(2)
polyFrac	14	14	16	16	16	15	(1)

Table 16: Comparative performance of the initial neural network with the polyFrac algorithm used in its structure.

Metrics	Initial NN	Initial NN using polyFrac
MAE	1.05	**0.86**
MAPE	4.2%	**3.1%**
RMSE	3.43	**2.78**

Table 17: Average error on 100D functions for swarm intelligence algorithms (CEC 2017 functions)

	HSSAPSO	**MPSO**	**HFPSO**	**PPSO**	**POLYFRAC**
F_1	6.76E+10	5.62E+10	4.44E+10	1.08E+04	**1.00E-42**
F_2	1.35E+04	3.76E+139	1.05E+133	2.22E+44	**1.23E-04**
F_3	1.08E+03	5.56E+05	5.01E+05	7.25E+04	**1.02E-44**
F_4	1.10E+03	7.37E+03	4.90E+03	2.40E+02	**1.30E+01**
F_5	629.77	1.86E+03	1.65E+03	5.20E+02	**3.41E+00**
F_6	2.03E+03	6.88E+02	6.82E+02	4.01E+01	**0.00E+00**
F_7	1.02E+03	3.04E+03	2.65E+03	7.38E+02	**4.03E+00**
F_8	1.13E+04	2.17E+03	2.01E+03	5.82E+02	**1.10E+00**
F_9	1.07E+04	7.37E+04	7.02E+04	1.48E+04	**1.00E-42**
F_{10}	4.92E+04	3.24E+04	2.76E+04	1.14E+04	**2.43E+03**
F_{11}	1.17E+06	1.50E+05	1.33E+05	8.96E+02	**1.00E-01**
F_{12}	6.82E+04	1.29E+10	5.38E+09	4.12E+06	**1.00E+03**
F_{13}	1.30E+06	1.50E+09	2.24E+08	1.53E+03	**1.00E+01**
F_{14}	1.00E+06	2.08E+07	8.74E+06	3.14E+05	**1.43E+02**
F_{15}	5.75E+03	4.99E+08	9.45E+06	4.69E+02	**3.30E+01**
F_{16}	2.10E+04	1.16E+04	8.02E+03	3.18E+03	**2.12E+02**
F_{17}	4.38E+06	9.30E+03	6.22E+03	2.63E+03	**8.44E+01**
F_{18}	1.24E+08	3.10E+07	1.12E+07	6.89E+05	**1.93E+02**
F_{19}	4.48E+03	5.00E+08	5.79E+07	5.42E+02	**1.91E+01**
F_{20}	3.15E+03	7.85E+03	6.41E+03	2.35E+03	**1.82E+02**
F_{21}	2.05E+04	3.86E+03	3.60E+03	1.06E+03	**1.22E+02**
F_{22}	4.04E+03	3.52E+04	3.11E+04	1.37E+04	**1.60E+03**
F_{23}	4.01E+03	4.70E+03	4.61E+03	2.07E+03	**2.22E+02**
F_{24}	3.1203e+03	5.22E+03	5.39E+03	**1.89E+03**	3.91E+03
F_{25}	2.0292e+04	8.56E+03	6.35E+03	**7.59E+02**	6.01E+03
F_{26}	3.3799e+03	2.29E+04	2.36E+04	1.56E+04	**1.12E+02**
F_{27}	1.0671e+04	4.15E+03	4.15E+03	1.34E+03	**1.63E+02**
F_{28}	1.1143e+04	1.08E+04	8.78E+03	5.86E+02	**4.80E+02**
F_{29}	1.4591e+05	1.38E+04	1.09E+04	3.73E+03	**3.62E+03**

[3] Y. Meraihi, A. Ramdane-Cherif, D. Acheli, M. Mahseur, Dragonfly algorithm: a comprehensive review and applications, Neural Computing & Applications (2020) 16625–16646.

[4] M. A. Al-Betar, Z. A. A. Alyasseri, M. A. Awadallah, I. A. Doush, Coronavirus herd immunity optimizer (chio).

[5] I. Goodfellow, Y. Bengio, A. Courville, Deep learning, MIT press, 2016.

[6] A. Al-Dujaili, S. Suresh, N. Sundararajan, Mso: a framework for bound-constrained black-box global optimization algorithms, Journal of Global Optimization 66 (4) (2016) 811–845.

[7] D. R. Jones, C. D. Perttunen, B. E. Stuckman, Lipschitzian optimization without the lipschitz constant, Journal of optimization Theory and Applications 79 (1) (1993) 157–181.

[8] M. Demirhan, L. Özdamar, L. Helvacıoğlu, Ş. I. Birbil, Fractop: A geometric partitioning metaheuristic for global optimization, Journal of Global Optimization 14 (4) (1999) 415–436.

[9] S. Kirkpatrick, Optimization by simulated annealing: Quantitative studies, Journal of statistical physics 34 (5-6) (1984) 975–986.

[10] D. Ashlock, J. Schonfeld, A fractal representation for real optimization, in: 2007 IEEE Congress on Evolutionary Computation, IEEE, 2007, pp. 87–94.

[11] A. Nakib, S. Ouchraa, N. Shvai, L. Souquet, E.-G. Talbi, Deterministic metaheuristic based on fractal decomposition for large-scale optimization, Applied Soft Computing 61 (2017) 468–485.

[12] M. A. Awadallah, M. A. Al-Betar, A. L. Bolaji, I. A. Doush, A. I. Hammouri, M. Mafarja, Island artificial bee colony for global optimization, Soft Computing (2020) 1–27.

[13] D. Karaboga, An idea based on honey bee swarm for numerical optimization, Tech. rep., Technical report-tr06, Erciyes university, engineering faculty (2005).

[14] R. V. Rao, A. Saroj, A self-adaptive multi-population based jaya algorithm for engineering optimization, Swarm and Evolutionary computation 37 (2017) 1–26.

[15] N. Bacanin, T. Bezdan, E. Tuba, I. Strumberger, M. Tuba, Monarch butterfly optimization based convolutional neural network design, Mathematics 8 (6) (2020) 936.

[16] Y. G. Petalas, K. E. Parsopoulos, M. N. Vrahatis, Memetic particle swarm optimization, Annals of operations research 156 (1) (2007) 99–127.

[17] I. B. Aydilek, A hybrid firefly and particle swarm optimization algorithm for computationally expensive numerical problems, Applied Soft Computing 66 (2018) 232–249.

[18] A. Tangherloni, L. Rundo, M. S. Nobile, Proactive particles in swarm optimization: A settings-free algorithm for real-parameter single objective optimization problems, in: 2017 IEEE Congress on Evolutionary Computation (CEC), IEEE, 2017, pp. 1940–1947.

[19] N. Singh, S. Singh, E. H. Houssein, Hybridizing salp swarm algorithm with particle swarm optimization algorithm for recent optimization functions, Evolutionary Intelligence (2020) 1–34.

[20] M. A. Potter, K. A. De Jong, A cooperative coevolutionary approach to function optimization, in: International Conference on Parallel Problem Solving from Nature, Springer, 1994, pp.

Table 18: The frequency at which the global optimum is reached for other metaheuristics.

polyFrac vs.	$50D$	**Rank**
IACO $\mathbb{R}$-Hybrid	14	(1)
2S-Ensemble	5	(4)
MOS-CEC2013	4	(5)
MOS-CEC2012	1	(6)
MOS-SOCO2011	14	(1)

249–257.

[21] F. Van den Bergh, A. P. Engelbrecht, A cooperative approach to particle swarm optimization, IEEE transactions on evolutionary computation 8 (3) (2004) 225–239.

[22] F. Van den Bergh, An Analysis of Particle Swarm Optimizers, Ph.D. dissertation, Dept. Comput. Sci., Univ. Pretoria, 2002.

[23] Z. Yang, K. Tang, X. Yao, Large scale evolutionary optimization using cooperative coevolution, Information sciences 178 (15) (2008) 2985–2999.

[24] Y.-j. Shi, H.-f. Teng, Z.-q. Li, Cooperative co-evolutionary differential evolution for function optimization, in: International Conference on Natural Computation, Springer, 2005, pp. 1080–1088.

[25] A. Auger, N. Hansen, A restart cma evolution strategy with increasing population size, in: 2005 IEEE congress on evolutionary computation, Vol. 2, IEEE, 2005, pp. 1769–1776.

[26] N. Hansen, A. Auger, R. Ros, S. Finck, P. Pošík, Comparing results of 31 algorithms from the black-box optimization benchmarking bbob-2009, in: Proceedings of the 12th annual conference companion on Genetic and evolutionary computation, 2010, pp. 1689–1696.

[27] S. Lloyd, Least squares quantization in pcm, IEEE transactions on information theory 28 (2) (1982) 129–137.

[28] A. H. Land, A. G. Doig, An automatic method for solving discrete programming problems, in: 50 Years of Integer Programming 1958-2008, Springer, 2010, pp. 105–132.

[29] F. Herrera, M. Lozano, D. Molina, Test suite for the special issue of soft computing on scalability of evolutionary algorithms and other metaheuristics for large scale continuous optimization problems, Last accessed: July.

[30] R. Cheng, M. Li, Y. Tian, X. Zhang, S. Yang, Y. Jin, X. Yao, Benchmark functions for cec'2017 competition on evolutionary many-objective optimization, School Comput. Sci., Univ. Birmingham, Birmingham, UK, Rep. CSR-17-01.

[31] M. Lozano, D. Molina, F. Herrera, Editorial scalability of evolutionary algorithms and other metaheuristics for large-scale continuous optimization problems, Soft Computing 15 (11) (2011) 2085–2087.

[32] R. Storn, K. Price, Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces, Journal of global optimization 11 (4) (1997) 341–359.

[33] L. J. Eshelman, J. D. Schaffer, Real-coded genetic algorithms and interval-schemata, in: Foundations of genetic algorithms, Vol. 2, Elsevier, 1993, pp. 187–202.

[34] D. Molina, M. Lozano, A. M. Sánchez, F. Herrera, Memetic algorithms based on local search chains for large scale continuous optimisation problems: Ma-ssw-chains, Soft Computing 15 (11) (2011) 2201–2220.

[35] A. LaTorre, S. Muelas, J.-M. Peña, A mos-based dynamic memetic differential evolution algorithm for continuous optimization: a scalability test, Soft Computing 15 (11) (2011) 2187–2199.

[36] L.-Y. Tseng, C. Chen, Multiple trajectory search for large scale global optimization, in: 2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence), IEEE, 2008, pp. 3052–3059.

[37] A. K. Qin, P. N. Suganthan, Self-adaptive differential evolution algorithm for numerical optimization, in: 2005 IEEE congress on evolutionary computation, Vol. 2, IEEE, 2005, pp. 1785–1791.

[38] M. Z. Ali, N. H. Awad, P. N. Suganthan, Multi-population differential evolution with balanced ensemble of mutation strategies for large-scale global optimization, Applied Soft Computing 33 (2015) 304–327.

[39] J. Brest, M. S. Maučec, Self-adaptive differential evolution algorithm using population size reduction and three strategies, Soft Computing 15 (11) (2011) 2157–2174.

[40] A. LaTorre, S. Muelas, J.-M. Peña, A comprehensive comparison of large scale global optimizers, Information Sciences 316 (2015) 517–549.

[41] A. W. Mohamed, A. A. Hadi, A. M. Fattouh, K. M. Jambi, Lshade with semi-parameter adaptation hybrid with cma-es for solving cec 2017 benchmark problems, in: 2017 IEEE Congress on evolutionary computation (CEC), IEEE, 2017, pp. 145–152.

[42] J. Brest, M. S. Maučec, B. Bošković, Single objective real-parameter optimization: Algorithm jso, in: 2017 IEEE congress on evolutionary computation (CEC), IEEE, 2017, pp. 1311–1318.

[43] K. M. Sallam, S. M. Elsayed, R. A. Sarker, D. L. Essam, Multi-method based orthogonal experimental design algorithm for solving cec2017 competition problems, in: 2017 IEEE Congress on Evolutionary Computation (CEC), IEEE, 2017, pp. 1350–1357.

[44] A. LaTorre, J.-M. Peña, A comparison of three large-scale global optimizers on the cec 2017 single objective real parameter numerical optimization benchmark, in: 2017 IEEE Congress on Evolutionary Computation (CEC), IEEE, 2017, pp. 1063–1070.

[45] D. Maharana, R. Kommadath, P. Kotecha, Dynamic yin-yang pair optimization and its performance on single objective real parameter problems of cec 2017, in: 2017 IEEE Congress on Evolutionary Computation (CEC), IEEE, 2017, pp. 2390–2396.

[46] R. Biedrzycki, A version of ipop-cma-es algorithm with midpoint for cec 2017 single objective bound constrained problems, in: 2017 IEEE Congress on Evolutionary Computation (CEC), IEEE, 2017, pp. 1489–1494.

[47] R. Kommadath, P. Kotecha, Teaching learning based optimization with focused learning and its performance on cec2017 functions, in: 2017 IEEE congress on evolutionary computation (CEC), IEEE, 2017, pp. 2397–2403.

[48] D. M. Cabrera, Taco, Association for Computing Machinery, https://tacolab.org/.

[49] G. Khodabandelou, W. Kheriji, F. H. Selem, Link traffic speed forecasting using convolutional attention-based gated recurrent unit, Applied Intelligence (2020) 1–22.

[50] G. Khodabandelou, M. Katranji, S. Kraiem, W. Kheriji, F. HadjSelem, Attention-based gated recurrent unit for links traffic speed forecasting, in: 2019 IEEE Intelligent Transportation Systems Conference (ITSC), IEEE, 2019, pp. 2577–2583.

Table 19: First column: Friedman Rank sum of other algorithms at dimension 50D. Second and third column: Raw and adjusted ρ-values from the Wilcoxon test for other metaheuristics using the Holm procedure.

polyFrac vs.	50D	50D	
		Raw	Adjusted
MOS-SOCO2011	2.578947368 (4)*	**4.89E-01**†	**7.11E-01**†
MOS-CEC2013	4.421052632 (6)	1.94E-07	6.01E-07
MOS-CEC2012	5.236842105 (7)	3.17E-08	1.54E-08
IACO $\mathbb{R}$-Hybrid	2.236842105 (2)	**5.46E-01**†	**5.75E-01**†
2S-Ensemble	4.157894737 (5)	2.04E-05	1.43E-04
FDA	2.368421053 (3)	1.32E-07	2.06E-06

* *the rank of algorithm with respect to the others.*

† *ρ-value > 0.05 fails to indicate the difference with significant level α = 0.05*

Table 20: Average error on 50D functions.

	MA-SSW-Chains	**jDElscop**	**CHC**	**mDE-bES**	**DE**	**MTS-LS1**	**SaDE**	**FDA**	**polyFrac**
F_1	**0.00E+00**	**0.00E+00**	1.67E-11	**0.00E+00**	**0.00E+00**	**0.00E+00**	2.68E+01	**0.00E+00**	**0.00E+00**
F_2	7.61E-02	3.15E-02	6.19E+01	1.52E+01	8.84E-11	8.84E-14	1.21E+02	2.71E-12	**1.33E-14**
F_3	4.79E+01	2.28E+01	1.25E+06	**4.76E-05**	1.63E+02	1.63E+02	7.46E+04	9.32E+01	5.90E-01
F_4	1.19E-01	**0.00E+00**	7.43E+01	1.77E+01	**0.00E+00**	**0.00E+00**	1.07E+01	**0.00E+00**	**0.00E+00**
F_5	**0.00E+00**	**0.00E+00**	1.67E-03	**0.00E+00**	7.68E-03	7.68E-03	1.87E-01	**0.00E+00**	**0.00E+00**
F_6	4.89E-14	**9.55E-14**	6.15E-07	3.97E-14	0.00E+00	0.00E+00	4.63E-02	6.75E-14	5.01E-14
F_7	**0.00E+00**	**0.00E+00**	2.66E-09	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
F_8	3.06E-01	9.97E-03	2.24E+02	1.64E-09	9.56E-12	9.65E-12	6.92E+05	**0.00E+00**	**0.00E+00**
F_9	2.94E+02	**0.00E+00**	3.10E+02	**0.00E+00**	1.03E+02	1.03E+02	3.00E-02	**0.00E+00**	**0.00E+00**
F_{10}	**0.00E+00**	**0.00E+00**	7.30E+00	**0.00E+00**	**0.00E+00**	**0.00E+00**	2.94E-02	**0.00E+00**	**0.00E+00**
F_{11}	4.49E-03	**0.00E+00**	2.16E+00	1.15E-08	1.04E+02	1.04E+02	8.35E-02	**0.00E+00**	**0.00E+00**
F_{12}	**0.00E+00**	**0.00E+00**	9.57E-01	0.00E+00	1.34E+01	1.34E+01	4.80E+01	**0.00E+00**	**0.00E+00**
F_{13}	3.02E+01	1.36E+01	2.08E+06	2.50E-01	2.94E+01	2.94E+01	3.42E+09	5.50E+01	**7.81E-14**
F_{14}	**0.00E+00**	**0.00E+00**	6.17E+01	9.60E+00	5.52E+01	5.52E+01	4.22E+03	**0.00E+00**	**0.00E+00**
F_{15}	**0.00E+00**	**0.00E+00**	3.98E-01	**0.00E+00**	**0.00E+00**	**0.00E+00**	8.50E-03	**0.00E+00**	**0.00E+00**
F_{16}	4.06E-03	0.00E+00	2.95E-09	**0.00E+00**	4.06E+01	4.06E+01	1.36E+01	**0.00E+00**	**0.00E+00**
F_{17}	2.60E+01	7.43E-03	2.26E+04	2.42E-01	2.17E+02	2.17E+02	2.36E+05	**6.09E-04**	5.90E-01
F_{18}	**0.00E+00**	2.41E-14	1.58E+01	5.65E-05	5.65E+01	5.65E+01	2.72E+01	**0.00E+00**	**0.00E+00**
F_{19}	**0.00E+00**	**0.00E+00**	3.59E+02	**0.00E+00**	**0.00E+00**	**0.00E+00**	1.15E-01	**0.00E+00**	**0.00E+00**

Table 21: Average error on 100D functions.

	MA-SSW-Chains	**jDElscop**	**CHC**	**mDE-bES**	**DE**	**MTS-LS1**	**SaDE**	**FDA**	**polyFrac**
F_1	**0.00E+00**	**0.00E+00**	3.56E-11	**0.00E+00**	3.79E+00	1.09E-12	3.13E+01	**0.00E+00**	**0.00E+00**
F_2	7.01E+00	1.21E+00	8.58E+01	4.00E+01	7.58E+01	4.66E-10	1.26E+02	8.48E-12	**4.30E-14**
F_3	1.38E+02	6.13E+01	4.19E+06	4.90E-01	1.27E+02	2.32E+02	1.11E+05	5.09E+01	**6.09E-02**
F_4	1.19E-01	**0.00E+00**	2.19E+02	1.87E+01	2.85E+00	1.05E-12	1.58E+01	**0.00E+00**	**0.00E+00**
F_5	**0.00E+00**	**0.00E+00**	3.83E-03	**0.00E+00**	3.05E-01	6.70E-03	3.53E-01	**0.00E+00**	**0.00E+00**
F_6	**6.03E-14**	2.00E-13	4.10E-07	1.44E-13	4.34E-01	1.20E-12	8.32E-02	1.35E-13	2.06E-14
F_7	**0.00E+00**	**0.00E+00**	1.40E-02	**0.00E+00**	**0.00E+00**	**0.00E+00**	0.00E+00	**0.00E+00**	**0.00E+00**
F_8	3.48E+01	5.57E+00	1.69E+03	2.32E-03	4.74E+02	1.43E-03	2.83E+05	**0.00E+00**	**0.00E+00**
F_9	5.63E+02	7.18E-09	5.86E+02	**0.00E+00**	3.71E-03	2.20E+02	3.00E-02	**0.00E+00**	**0.00E+00**
F_{10}	**0.00E+00**	**0.00E+00**	3.30E+01	**0.00E+00**	**0.00E+00**	**0.00E+00**	4.73E-02	**0.00E+00**	**0.00E+00**
F_{11}	1.09E-01	8.17E-09	7.32E+01	0.00E+00	8.58E-04	2.10E+02	3.05E-01	**0.00E+00**	**0.00E+00**
F_{12}	3.28E-03	**0.00E+00**	1.03E+01	5.36E-04	2.71E+00	3.91E+01	3.79E+01	**0.00E+00**	**0.00E+00**
F_{13}	8.35E+01	5.11E+01	2.70E+06	8.50E+00	5.87E+01	1.75E+02	3.42E+09	1.68E+02	**6.09E-02**
F_{14}	**0.00E+00**	**0.00E+00**	1.66E+02	1.16E+01	2.21E+00	2.04E+02	3.92E+03	**0.00E+00**	**0.00E+00**
F_{15}	**0.00E+00**	**0.00E+00**	8.13E+00	**0.00E+00**	**0.00E+00**	**0.00E+00**	3.99E-02	**0.00E+00**	**0.00E+00**
F_{16}	1.61E-02	**0.00E+00**	2.23E+01	**0.00E+00**	3.52E+00	1.04E+02	1.96E+01	**0.00E+00**	**0.00E+00**
F_{17}	9.92E+01	3.21E-01	1.47E+05	**6.65E-03**	1.58E+01	4.17E+02	2.34E+05	6.22E+00	6.09E-02
F_{18}	**0.00E+00**	6.33E-14	7.00E+01	4.46E-01	8.76E-01	1.22E+02	3.05E+01	**0.00E+00**	**0.00E+00**
F_{19}	**0.00E+00**	**0.00E+00**	5.45E+02	**0.00E+00**	**0.00E+00**	**0.00E+00**	2.71E-01	**0.00E+00**	**0.00E+00**

Table 22: Average error on 200D functions.

	MA-SSW-CHAINS	JDELSCOP	CHC	MDE-BES	DE	MTS-LS1	SADE	FDA	POLYFRAC
F_1	**0.00E+00**	**0.00E+00**	8.34E-01	**0.00E+00**	8.55E+00	2.29E+00	2.03E+01	**0.00E+00**	**0.00E+00**
F_2	3.36E+01	7.54E+00	1.03E+02	4.15E+01	1.05E+02	4.54E-09	1.03E+02	1.23E-10	**1.02E-11**
F_3	2.50E+02	1.40E+02	2.01E+07	1.35E+02	3.32E+05	1.69E+02	4.82E+04	2.51E+02	**0.00E+00**
F_4	4.43E+00	**0.00E+00**	5.40E+02	9.27E-13	6.98E+00	2.34E-12	6.25E+00	**0.00E+00**	**0.00E+00**
F_5	**0.00E+00**	**0.00E+00**	8.76E-03	**0.00E+00**	4.05E-01	5.42E-03	6.43E-02	**0.00E+00**	**0.00E+00**
F_6	1.19E-13	4.52E-13	1.23E+00	**0.00E+00**	7.14E-01	2.38E-12	2.73E-02	2.52E-13	1.62E-13
F_7	**0.00E+00**	**0.00E+00**	2.59E-01	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
F_8	7.23E+02	2.52E+02	9.38E+03	8.71E-01	5.76E+03	1.42E+01	4.47E+05	**0.00E+00**	**0.00E+00**
F_9	1.17E+03	4.30E-08	1.19E+03	**0.00E+00**	8.79E-03	4.27E+02	3.00E-02	**0.00E+00**	**0.00E+00**
F_{10}	**0.00E+00**	**0.00E+00**	7.13E+01	**0.00E+00**	4.19E-02	**0.00E+00**	1.59E-02	**0.00E+00**	**0.00E+00**
F_{11}	3.50E-01	9.58E-09	3.85E+02	**0.00E+00**	5.07E-03	4.28E+02	4.89E-03	**0.00E+00**	**0.00E+00**
F_{12}	1.75E-02	**0.00E+00**	7.44E+01	**0.00E+00**	3.61E+00	8.42E+01	4.63E+01	**0.00E+00**	**0.00E+00**
F_{13}	1.68E+02	1.10E+02	5.75E+06	9.45E+01	1.49E+02	2.53E+02	3.16E+09	7.07E+01	**3.59E-13**
F_{14}	9.76E-01	4.11E-16	4.29E+02	1.20E+01	4.75E+00	3.98E+02	4.09E+03	**0.00E+00**	**0.00E+00**
F_{15}	**0.00E+00**	**0.00E+00**	2.14E+01	**0.00E+00**	**0.00E+00**	**0.00E+00**	5.38E-03	**0.00E+00**	**0.00E+00**
F_{16}	6.02E-02	**0.00E+00**	1.60E+02	**0.00E+00**	3.70E+00	1.97E+02	9.49E+00	**0.00E+00**	**0.00E+00**
F_{17}	7.55E+01	2.39E+01	1.75E+05	8.39E-02	2.23E+01	6.07E+02	2.36E+05	9.31E+01	**0.00E+00**
F_{18}	4.29E-04	2.04E-13	2.12E+02	8.93E-11	2.37E+00	2.34E+02	1.69E+01	**0.00E+00**	**0.00E+00**
F_{19}	**0.00E+00**	**0.00E+00**	2.06E+03	**0.00E+00**	4.19E-02	**0.00E+00**	1.00E-01	**0.00E+00**	**0.00E+00**

Table 23: Average error on 500D functions.

	MA-SSW-CHAINS	JDELSCOP	CHC	MDE-BES	DE	MTS-LS1	SADE	FDA	POLYFRAC
F_1	**0.00E+00**	**0.00E+00**	2.84E-12	3.92E-13	2.46E+01	5.77E-12	1.34E+01	**0.00E+00**	**0.00E+00**
F_2	7.86E+01	3.06E+01	1.29E+02	4.56E+01	1.44E+02	5.34E-06	9.23E+01	4.30E-04	**9.00E-10**
F_3	6.07E+02	4.06E+02	1.14E+06	4.16E+02	1.12E+05	2.20E+02	2.62E+04	5.82E+02	**0.00E+00**
F_4	1.78E+02	1.59E-01	1.91E+03	1.91E-11	1.63E+01	5.62E-12	1.31E+00	**0.00E+00**	**0.00E+00**
F_5	**0.00E+00**	**0.00E+00**	6.98E-03	1.83E-13	4.73E-01	4.24E-03	7.48E-03	**0.00E+00**	**0.00E+00**
F_6	2.63E-13	1.18E-12	5.16E+00	**3.56E-14**	1.06E+00	6.18E-12	4.63E-01	8.74E-13	7.27E-13
F_7	4.69E-14	**0.00E+00**	1.27E-01	**0.00E+00**	**0.00E+00**	1.46E-12	**0.00E+00**	**0.00E+00**	**0.00E+00**
F_8	1.32E+04	5.66E+03	7.22E+04	5.48E+02	6.70E+04	6.16E+03	3.21E+05	**0.00E+00**	**0.00E+00**
F_9	2.53E+03	6.10E-08	3.00E+03	**0.00E+00**	1.12E-02	1.00E+03	3.00E-02	**0.00E+00**	**0.00E+00**
F_{10}	2.80E-01	**0.00E+00**	1.86E+02	**0.00E+00**	2.93E-01	**0.00E+00**	8.41E-03	**0.00E+00**	**0.00E+00**
F_{11}	4.21E+01	4.40E-08	1.81E+03	**0.00E+00**	2.43E-01	1.00E+03	2.22E-03	**0.00E+00**	**0.00E+00**
F_{12}	2.55E+01	**0.00E+00**	4.48E+02	**0.00E+00**	1.16E+01	2.47E+02	4.61E+01	**0.00E+00**	**0.00E+00**
F_{13}	4.00E+02	3.14E+02	3.22E+07	3.23E+02	4.02E+02	5.05E+02	2.97E+09	3.74E+02	**9.02E-13**
F_{14}	5.65E+01	8.00E-02	1.46E+03	1.68E+01	1.16E+01	1.10E+03	3.91E+03	**0.00E+00**	**0.00E+00**
F_{15}	5.53E+00	**0.00E+00**	6.01E+01	**0.00E+00**	4.19E-02	1.08E-12	2.84E-03	**0.00E+00**	**0.00E+00**
F_{16}	1.08E-01	**0.00E+00**	9.55E+02	**0.00E+00**	1.32E+01	4.99E+02	5.82E+00	**0.00E+00**	**0.00E+00**
F_{17}	1.38E+02	7.65E+01	8.40E+05	6.65E+01	6.94E+01	7.98E+02	2.38E+05	3.96E+02	**0.00E+00**
F_{18}	2.41E-03	1.11E-12	7.32E+02	**0.00E+00**	3.87E+00	5.95E+02	9.43E+00	**0.00E+00**	**0.00E+00**
F_{19}	**0.00E+00**	**0.00E+00**	1.76E+03	**0.00E+00**	8.39E-02	**0.00E+00**	1.00E-01	**0.00E+00**	**0.00E+00**

Table 24: Average error on 1000D functions.

	MA-SSW-CHAINS	JDELSCOP	CHC	MDE-BES	DE	MTS-LS1	SADE	FDA	POLYFRAC
F_1	**0.00E+00**	**0.00E+00**	1.36E-11	8.24E-13	3.71E+01	1.15E-11	3.49E+01	**0.00E+00**	**0.00E+00**
F_2	1.39E+02	6.14E+01	1.44E+02	5.97E+01	1.63E+02	2.25E-02	1.43E+02	3.11E-01	**0.00E+00**
F_3	1.22E+03	8.48E+02	8.75E+03	9.00E+02	1.59E+05	2.10E+02	1.62E+05	1.13E+03	**1.97E-12**
F_4	1.58E+03	1.99E-01	4.76E+03	4.03E+01	3.47E+01	1.15E-11	3.21E+01	**0.00E+00**	**0.00E+00**
F_5	5.92E-04	**0.00E+00**	7.02E-03	**0.00E+00**	7.36E-01	3.55E-03	6.33E-01	**0.00E+00**	**0.00E+00**
F_6	1.46E-09	2.67E-12	1.38E+01	1.28E-12	8.70E-01	1.24E-11	4.28E-01	1.91E-12	**0.00E+00**
F_7	6.23E-13	**0.00E+00**	3.52E-01	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
F_8	7.49E+04	3.21E+04	3.11E+05	7.98E+03	3.15E+05	1.23E+05	3.08E+05	**0.00E+00**	**0.00E+00**
F_9	5.99E+03	4.40E-03	6.11E+03	**0.00E+00**	6.26E-02	1.99E+03	3.00E-02	**0.00E+00**	**0.00E+00**
F_{10}	2.09E-05	**0.00E+00**	3.83E+02	**0.00E+00**	1.67E-01	**0.00E+00**	1.47E-01	**0.00E+00**	**0.00E+00**
F_{11}	5.27E+01	8.58E-04	4.82E+03	**0.00E+00**	4.42E-02	1.99E+03	4.56E-01	**0.00E+00**	9.46E-05
F_{12}	9.48E-02	**0.00E+00**	1.05E+03	**0.00E+00**	2.58E+01	5.02E+02	3.43E+01	**0.00E+00**	**0.00E+00**
F_{13}	1.02E+03	6.57E+02	6.66E+07	6.34E+02	8.24E+04	8.87E+02	3.27E+09	7.69E+02	**0.00E+00**
F_{14}	7.33E+02	3.98E-02	3.62E+03	2.45E+01	2.39E+01	2.23E+03	3.71E+03	**0.00E+00**	**0.00E+00**
F_{15}	1.16E-13	**0.00E+00**	8.37E+01	**0.00E+00**	2.11E-01	**0.00E+00**	1.11E-01	**0.00E+00**	9.27E-13
F_{16}	2.19E+00	8.04E-01	2.32E+03	**0.00E+00**	1.83E+01	1.00E+03	2.37E+01	**0.00E+00**	**0.00E+00**
F_{17}	3.26E+02	1.72E+02	2.04E+07	1.88E+02	1.76E+05	1.56E+03	1.62E+05	1.95E+02	**0.00E+00**
F_{18}	2.58E+01	1.65E-01	1.72E+03	2.49E-01	7.55E+00	1.21E+03	3.54E+01	**0.00E+00**	**0.00E+00**
F_{19}	**0.00E+00**	**0.00E+00**	4.20E+03	**0.00E+00**	2.51E-01	**0.00E+00**	9.32E+02	**0.00E+00**	**0.00E+00**

Table 25: Average error on 50D functions for other metaheuristics.

	MOS-SOCO2011	MOS-CEC2013	MOS-CEC2012	IACO $\mathbb{R}$-HYBRID	2S-ENSEMBLE	FDA	POLYFRAC
F_1	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
F_2	5.88E-01	1.10E+02	1.03E+02	**0.00E+00**	4.31E+01	2.71E-12	1.33E-14
F_3	7.09E+01	7.39E+00	9.38E+02	**0.00E+00**	1.34E+03	9.32E+01	5.90E-01
F_4	**0.00E+00**	**0.00E+00**	1.90E+02	**0.00E+00**	8.58E-01	**0.00E+00**	**0.00E+00**
F_5	**0.00E+00**	**0.00E+00**	1.18E-03	**0.00E+00**	3.00E-03	**0.00E+00**	**0.00E+00**
F_6	**0.00E+00**	**0.00E+00**	1.03E+00	**0.00E+00**	**0.00E+00**	6.75E-14	5.01E-14
F_7	**0.00E+00**	2.56E-12	1.03E-13	**0.00E+00**	Inf.	**0.00E+00**	**0.00E+00**
F_8	1.66E+05	5.98E+03	1.09E+03	**0.00E+00**	1.93E+05	**0.00E+00**	**0.00E+00**
F_9	**0.00E+00**	2.51E+03	5.95E+03	**0.00E+00**	2.68E+00	**0.00E+00**	**0.00E+00**
F_{10}	**0.00E+00**	1.58E+00	1.79E+02	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
F_{11}	**0.00E+00**	2.54E+03	5.88E+03	**0.00E+00**	3.23E+00	**0.00E+00**	**0.00E+00**
F_{12}	**0.00E+00**	9.99E+02	1.12E+03	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
F_{13}	1.69E+02	1.23E+03	2.03E+03	8.77E-01	1.25E+03	5.50E+01	**7.81E-14**
F_{14}	**0.00E+00**	3.37E+03	4.32E+03	2.90E-02	4.40E-02	**0.00E+00**	**0.00E+00**
F_{15}	**0.00E+00**	1.93E-12	2.04E+01	**0.00E+00**	Inf.	**0.00E+00**	**0.00E+00**
F_{16}	**0.00E+00**	8.02E+03	2.33E+03	1.12E-03	**0.00E+00**	**0.00E+00**	**0.00E+00**
F_{17}	6.71E+01	3.55E+11	3.71E+03	**1.84E-06**	3.39E+01	6.09E-04	5.90E-01
F_{18}	**0.00E+00**	2.03E+03	2.29E+03	9.20E-01	5.51E-01	**0.00E+00**	**0.00E+00**
F_{19}	**0.00E+00**	2.05E+03	5.25E+01	**0.00E+00**	7.99E-17	**0.00E+00**	**0.00E+00**