Parallel fractal decomposition based algorithm for big continuous optimization problems

Amir Nakib, Léo Souquet, El-Ghazali Talbi

Version of Record: https://www.sciencedirect.com/science/article/pii/S0743731518304003
Manuscript_c68c9b997bf1316e4b8181ad2ae763ac

Parallel fractal decomposition based algorithm for big continuous optimization problems

A. Nakib[a], L. Souquet[a,b], and E-G. Talbi[c]

[a] *Université Paris-Est, Laboratoire LISSI,*
122 Rue Paul Armangot, 94400 Vitry sur Seine, France
[b] *Data ScienceTech Institute, DSTI Labs*
950 Route des Colles, Les Templiers, 061410 Biot, France
[c] *INRIA Lille - Nord Europe Parc Scientifique de la Haute Borne*
40, Avenue Halley, Bat A, Villeneuve d'Ascq, France

Abstract

Fractal Decomposition Algorithm (FDA) is a metaheuristic that was recently proposed to solve high dimensional continuous optimization problems. This approach is based on a geometric fractal decomposition which divide the search space while looking for the optimal solution. While FDA and its fractal decomposition has shown to be an effective optimization algorithm, its running time grows significantly as the problems dimension increases. To overcome this expensive computational time, a parallelized version of FDA, called *Parallel Fractal Decomposition Algorithm (PFDA)* is proposed. The focus was on parallelizing the exploration and exploitation phases of the original algorithm on a multi-threaded environment. The performances of PFDA were evaluated on the same Benchmark used to illustrate FDA efficiency, the SOCO 2011. It is composed of 19 functions with dimensions going from 50 to 5000. Results shows that PFDA reaches similar performances as the original version with a significantly reduced computational time.

Keywords: very-Large-scale optimization, Metaheuristics, Geometric Fractal Decomposition, Local Search. Continuous optimization.

1. Introduction

For a couple of years, massively parallel architectures are now available to a broader public, where it was before reserved for super-computers. The access to those parallel architectures through cloud platforms, grid computer allow to reduce the running time to solve complex problems. Algorithms need to be designed, implemented or modified to take profit from the power of all these new architectures. In other words, developers must either adapt an existing algorithm to benefit from the new resources or design new algorithms.

In the first approach, we can cite [1, 2, 3] where authors aimed to parallelize the well known mataheuristic *Simulated Annealing*. Other metaheuristics have also been modified to take benefit from Graphical Processing Units (GPU) such as ant colony optimization algorithm [4]. In the second approach, in [5] authors proposed a new parallelized metaheuristic based on Particle Swarm Optimization *(PSO)* principle, especially designed to be run on parallel architectures. For more detailed review on parallel metaheuristics reader can refer to [6].

Another approach to tackle global optimization problems is the Divide-and-Conquer (D&C) approach. It is efficient to solve small problems, however, due to the exponential complexity of these algorithms, they cannot be used for large scale problems. Recently, some works, inspired from the parallel work on branch and bound as in [7], were proposed to use this approach in large scale by using High Performance Computing (HPC) architectures.

As pointed out, adapting existing algorithms has been popular in the last decade. Generally, D&C based techniques are straightforward to parallelize, however, the challenge lies in achieving high performance and scalability in parallel. These algorithms divide the search space into sub-regions recursively, then, generate a search tree. In [7] the authors have implemented a (B&B) algorithm running on CPU, multi-GPU and/or heterogeneous environments. In a multi-CPU environment, a simple illustration would be to explore the different generated branches on different CPUs, in parallel.

In this work, the goal is to use the principle of the geometric fractal decom-

position to divide the search space. This new approach was proposed in [8] to solve large-scale continuous optimization problems. The algorithm called *Fractal Decomposition Algorithm (FDA)*, takes profit from a fractal decomposition and uses hyperspheres geometric form to divide the search space. Authors argue that the flexibility of the approach to cover the search space and its low complexity lead to this choice. Moreover, this choice was confirmed by the fact that FDA performs well on large-scale problems. Deterministic and single solution based, FDA can be seen as a Divide-and-Conquer approach, because it splits the search space using hyperspheres as an elementary geometric form, then, builds a search tree. While navigating through the tree FDA identifies, at each decomposition level, candidate optimal hyperspheres: areas where the global optimum could be found. This principle is illustrated in Figure 1 with in case of a four-level decomposition, the red hypersphere being the best one at each level. Once the maximum fractal depth k is reached, a local search is triggered to find the optimal solution. The performances of the algorithm were analyzed via a benchmark taken from the *special issue of soft computing on scalability of evolutionary algorithms (SOCO 2011)*. It consists in 19 large-scale optimization test functions with dimensions going from 50 to 1000. The obtained results were compared to other competing metaheuristics designed to solve similar problems as well as state-of-the-art algorithms.

In its original version, FDA was running on a mono-threaded environment and therefore its computational time increases significantly when the problems' dimension increases. The motivation of the current work was to address this issue. Reducing the execution time, solving big optimization problems (problems with dimensions higher than 1000), and maintaining the original precisions. We recall that as FDA is a D&C based approach, a search tree is built with hyperspheres. In this paper, a parallelized version of FDA, called PFDA, running on a multi-threaded environment using the framework OpenMP and following the $Fork/Join$ model is proposed. This approach is motivated by the fact that each thread can explore and exploit hyperspheres simultaneously. The aim is to significantly improve its running time with a focus on big optimization problems.

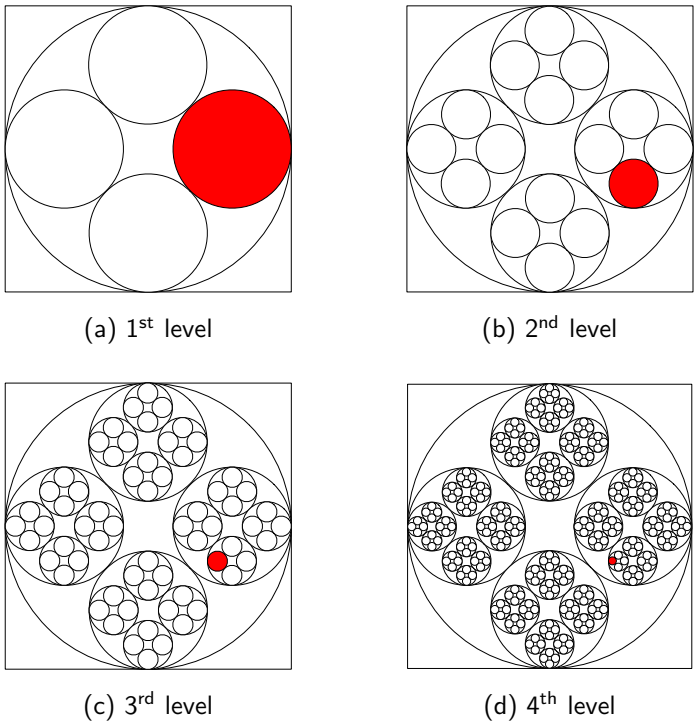


Figure 1: Illustration of the fractal decomposition of the search space: the depth of the decomposition is equal to 4.

The rest of this paper is organized as follow: in Section 2, the literature on parallelized metaheuristics is reviewed briefly. Section 3 recalls the Fractal Decomposition Algorithm. In Section 4 an analysis of the mono-threaded FDA is presented. Section 5 tackle the different challenges that parallelizing FDA. Section 6 illustrates and discusses the results obtained by the parallel implementation of the FDA algorithm. Finally, a conclusion ends the paper.

2. Related work

2.1. Parallelized metaheuristics

The parallelization of metaheuristics has been popular over the last three decades in the field of optimization. Indeed, several works have mainly focus on

adapting existing algorithms to allow them to take profit from multithreaded or multi-nodes environments.

For instance, *Simulated Annealing* was parallelized in 1987, to solve real life optimization problems [1, 2, 3]. Authors explored different strategies for running the algorithm on shared-memory multiprocessors by distributing the selection and evaluation parts to different processors in parallel. Similar optimization results (in terms of the cost function) as the serial version was obtained in shorter running time. In [3] the authors report an improvement of the speed by a factor of 6 on 8 processors.

In [12], the authors focused on parallelizing the well known metaheuristic called *Genetic Algorithm (GA)*. They developed a parallelized version, called *Parallel Genetic Algorithm (PGA)*. The main idea behind this algorithm is to distribute the selection scheme by making each individual looking for a good solution, but only among its neighbors. This approach allowed to obtain good results on the Travel Salesman problem.

Decades later, parallelizing metaheuristic is still a popular field with many other approaches being parallelized. The *Ant Colony Optimization (ACO)* algorithm has been the subject of many works [13, 14, 15, 16]. In their work [13], the authors run iteratively different sequential ACO [14], the parallelization is made at the colony or ant level, searching independently and sending back their results synchronously or asynchronously, depending on the parallel model. The latter has parallelized the algorithm on a multi-core processor environment using OpenMP. They concluded that the execution time can be greatly reduced without loosing quality of the final solution. In [16], authors studied a parallelized version of the ACO algorithm applied to the Taxi-Passenger Matching. The idea was to divide the city being optimized into several regions to reduce the dimension of the problem. Hence, making the approach similar to a D&C strategy. They explore regions in parallel allowing the algorithm to find a good solution faster.

2.2. Decomposition methods

As FDA is a Divide-and-Conquer based algorithm, then, the literature on Branch-and-Bound algorithms can also be take into account. In [17], the authors mentioned different strategies to parallelized B&B algorithms: 1. Parallelizing the nodes' evaluation; 2. Parallelizing the construction of the search tree; 3. A combination of the first two. They studied these three strategies on a multi-thread environment, reaching a linear tendency of the SpeedUp.

Only few algorithms that use the geometric decomposition of the search space were proposed in the literature: DIviding RECTangles (DIRECT) [18], FRACTOP [19] and Multiple Optima Sierpinski Searcher [20]. However, when dividing the search domain, both DIRECT and FRACTOP suffer from an exponential growth of subregions, making those algorithms computationally expensive on large-scale problems not applicable for big optimization. In the case of Multiple Optima Sierpinski Searcher, the authors stated that the chosen geometric forms will not allow the algorithm to cover the entire search domain. As DIRECT does not perform well on high dimensions problems, in [21], authors proposed a parallelized version of the algorithm to tackle this issue. To do so, multi-start strategy was used via evaluating multiple starting points on different processors. The evaluations of the objective function was also distributed among the different CPUs. This algorithm is implemented using both OpenMP for the multi-threading part and MPI for passing messages over multiple processors. It is known that DIRECT divides the search space into hypercubes, the number of vertices to evaluate grows exponentially, when the dimension of the problem increases making the algorithm computationally expensive on large-scale problems: seventeen (17) hours were necessary to reach 238397 function evaluations using 141 processors. It shows that even parallelized, DIRECT is not suited for large scale problems. FDA has been designed to address these two main issues that decomposition methods face, i.e not being able to cover the entire search domain and falling to solve large-scale and big optimization problems.

3. The Fractal Decomposition Algorithm: Recall

The *Fractal Decomposition Algorithm* [8] (FDA) a D&C based algorithm that has been designed to solve large-scale continuous optimization problems. While searching, FDA builds a search tree of promising optimum areas of a depth k (called fractal depth), by dividing the search space recursively using geometrical hyperspheres. The algorithm is composed of three main phases: 1. Initialization detailed in Sub-Section 3.2; 2. Exploration phase (in Sub-Section 3.3); 3. Exploitation Phase (in Sub-Section 3.4).

Algorithm 1 highlights the structure of the approach and Figure 4 shows the main life cycle using Unified Modeling Language (UML). For more details on FDA, the reader can refer to [8].

3.1. FDA Parameters

To be fine tuned FDA requires the setting of four different parameters, their values were taken from the original paper [8]. A recall of these values is given in the following:

- (k) = 5, the fractal depth;
- $\varphi = 0.5$, the coefficient by which the step-size is decreased, used in the *Intensive Local Search (ILS)* (Section 3.4);
- $\alpha = 1.75$, the relaxation coefficient used in the exploration procedure (section 3.3);
- $\omega_{min} = 10^{-20}$, the tolerance threshold, also used in *ILS* (section 3.4).

3.2. Initialization procedure

The first hypersphere, at level $l = 0$, is initialized at the center of the search space, and lies within its limits, as shown on Figure 2(a). It is the biggest hypersphere that can be created within the domain and both center $\vec{C}^{(1)}$ and radius r are computed using the expressions (1) and (2), respectively.

$$\vec{C}_j^{(1)} = L + (U - L)/2, \text{ for } j = 1, 2, \ldots, D \tag{1}$$

where $\vec{C}^{(1)}$ are the coordinates of the center of the biggest hypersphere within the search space, D the dimension of the search space and r the hypersphere's radius.

$$r = (U - L)/2 \tag{2}$$

where U is the upper bound, L is the lower bound of the whole search space.

Once the first hypersphere is created, it is partitioned into $2 \times D$ child-hyperspheres using the expression (3) and as shown on Figure 1a.

$$\vec{C}_k^{(i)} = \vec{C}_k^{(i)} + (-1)^i \times ((r - r') \times \vec{e}_k) \tag{3}$$

where $\vec{C}^{(i)}$ represents the center of the i_{th} child-hypersphere with $i = 1, \ldots, 2 \times D$, $r' = r/(1 + \sqrt{2})$ and $\vec{e}_k$ the unit vector at the dimension k.

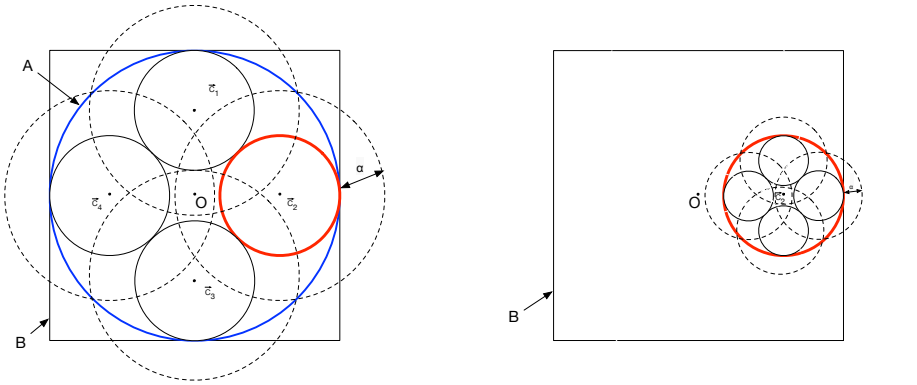
3.3. Exploration procedure

FDA uses a heuristic, called *promising hypersphere selection heuristic* for the exploration phase which is designed to detect the most promising hyperspheres to be further decomposed.

As the hyperspheres do not cover all the search space, FDA increases the radius of newly generated hyperspheres by a ratio α at the evaluation of the quality of the hyperspheres.

This procedure is called relaxation and is illustrated in Figure 2. It is applied to all $2 \times D$ child-hyperspheres. Their respective qualities are then computed and only the best one is selected to be decomposed, using expression (3), which leads the algorithm to move down one level in the search tree ($l = l + 1$). While evaluating hyperspheres, FDA keeps track of the best solution encountered and updates it after each hypersphere evaluation.

Hyperspheres that have not been decomposed are sorted according to their quality and stored in a stack for further decomposition. If all hyperspheres at a level l have been explored, FDA selects the next one in the stack at level $l = l - 1$ to be partitioned. This would have the effect of creating a new branch in the



(a) Geometric decomposition at level 1 (b) Geometric decomposition at level 2

Figure 2: Illustration of the decomposition procedure in the case of a 2D search space, where A is the biggest hypersphere inside the search space (B), C_1, C_2, C_3, and C_4 are centers of hyperspheres at the first level

search tree. FDA stops either when the stopping criterion is reached or when all the search tree has been explored (all branches of depth k are explored).

3.4. Exploitation procedure

For the current explored branch of the search tree, when the $k-th$ level is reached, FDA triggers a local search aiming to explore all generated child-hyperspheres in attempt to find better solution. At this step, different learning based optimization methods can be used. However, to satisfy the low complexity design constraint, a simple algorithm, called Intensive Local Search (ILS) was implemented.

For each hypersphere at the level k, ILS starts with $\vec{x}^s$ being initialized at the center $\vec{C}$ moving along each dimension and evaluating two solutions $\vec{x}^{s_1}$ and $\vec{x}^{s_2}$ as expressed in (4) and (5), respectively.

$$\vec{x}^{s_1} = \vec{x}^s + \omega \times \vec{e}_i \tag{4}$$

$$\vec{x}^{s_2} = \vec{x}^s - \omega \times \vec{e}_i \tag{5}$$

where $\vec{e_i}$ is the unit vector where the i^{th} element is set to 1, and other elements to 0. ω is the step size in which $\vec{e_i}$ changes.

Afterwards, the best solution among $\vec{x}^s$, $\vec{x}^{s_1}$ and $\vec{x}^{s_2}$ is chosen to be the next current solution $\vec{x}^s$ and ILS moves to the next dimension.

If no improvement has been made for $\vec{x}^s$, then ω is reduced by factor $1/\varphi$, Intensive Local Search stops when one of the following conditions is satisfied:

- Stopping criterion is reached.
- The step size reaches ω_{min}, the tolerance or the precision need of the problem being solved.

Once a hypersphere has been explored, ILS returns $\vec{x}^s$ containing the best solution found locally. Then, the best solution found so far is updated if it is worst than $\vec{x}^s$.

Once all hyperspheres at the $k-th$ level have been explored, if the stopping criterion has not been yet met, then, FDA backtracks in the search tree, and the next hypersphere to be decomposed is selected at the level $l = l - 1$. This procedure allows the algorithm to explore other regions of the search space.

To illustrate this behavior, the Figure 3 shows different levels explored by FDA when solving the shifted Griewank problem where $D = 5$. As it can be seen, once FDA explored all hyperspheres at the 5-th level, it selects the next one of the 4-th level to be decomposed, and triggers again ILS on each 5-th level hyperspheres. Once all hyperspheres at the 4-th level has been decomposed, it selects the next one from the 3-rd level, and continues until the stopping criterion is met.

Moreover, this illustration points out the fact that FDA explores the whole search space. In other terms, if none stopping criterion was set, FDA will stop when the entire search tree has been explored.

4. Analysis of the mono-thread implementation of FDA

The Figure 5 illustrates four main phases of FDA. Figure 5a represents the first hypersphere (in red) being decomposed into $2 \times D$ child-hyperspheres

Algorithm 1: FDA Algorithm

Input: Deep of the fractal decomposition: $k = 5$ and precision threshold: $\omega_{\min} = 1 \times e^{-20}$

Input: Coefficient step-size: $\lambda = 0.5$, inflation coefficient: $\alpha = 1.75$ and dimension of the problem: D

`// Initialization phase as described in Section 3.2`

Initialize the center $\vec{C}$ of the first Hypersphere, at the center of the search space using (1)

while *Stopping criterion is not reached* **do**

Partition the current hypersphere H using the Fractal procedure given in expression (3)

`// Exploration phase as in Section 3.3`

for $2 \times D$ *l-level hypersphere* **do**

Relaxe hypersphere by α and compute its quality

end

Sort the $2 \times D$ hyperspheres at the current l-level by their quality

Replace the current hypersphere H by the first of the sorted hyperspheres at the current level

if l==k **then**

`// last level reached`

`// Exploitation phase as in Section 3.4`

for $2 \times D$ *hypersphers at the last level* **do**

Apply the ILS heuristics on each created hypersphere

end

if *stopping criterion is not reached* **then**

Move up one level ($l = l - 1$)

end

else

Go to next level: $l = l + 1$

end

end

Result: the best solution *BestSol* and its coordinates

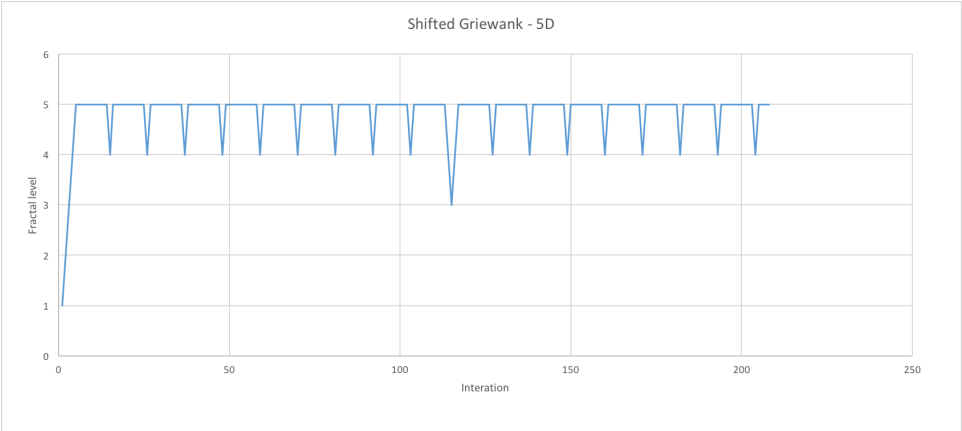


Figure 3: Illustration of the way FDA backtracks in the search tree in dimension $D = 5$

(CH_i). For more clarity, in this example the dimension is set to $D = 2$. It can be seen on Figure 5b that child-hyperspheres are evaluated sequentially. Once the child-hypersphere with the best quality is found (CH_2 colored in red in this case), then, it is also decomposed into $2 \times D$ child-hyperspheres (Figure 5c). When the depth k is reached *(k set to two in our example)*, ILS is triggered on all created child-hyperspheres. In Figure 5d one sphere is exploited by the heuristic at a time. When all $k-th$ level child-hyperspheres have been exploited, FDA either terminates if the stopping criterion has been reached or backtracks in the search tree and continues.

One can remark that both the exploration and exploitation phases handle hyperspheres sequentially and create bottlenecks.

5. Proposed Multi-threaded Implementation Strategy

Finding a good solution (if the optimum is not known) and within a reasonable time are the two main aspects to be taken into account when designing a metaheuristic. The increase in the complexity of the problem will naturally increase the computation time required for the algorithm to find the desired solution.

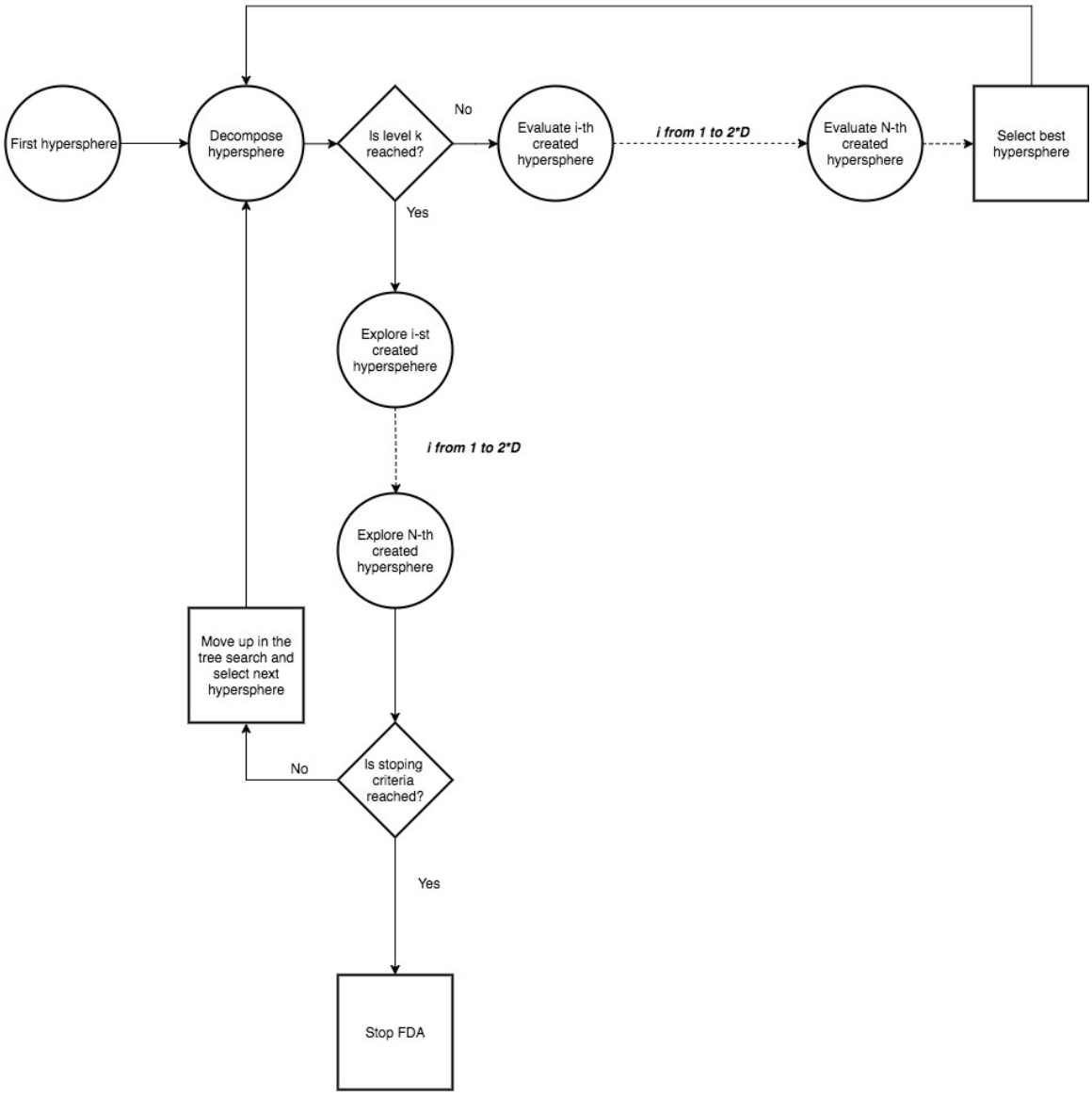
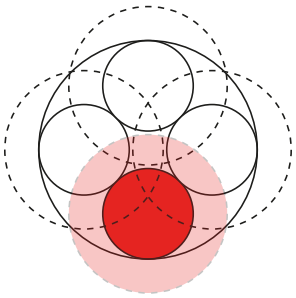
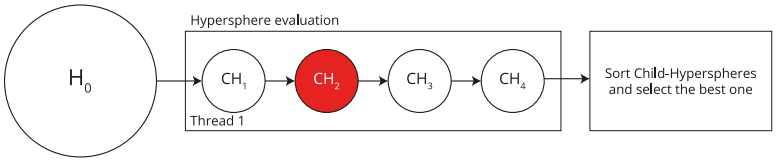


Figure 4: Illustration of life-cycle of the mono-threaded version of FDA.

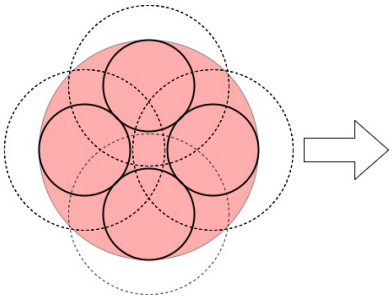
This section describes the proposed Parallel FDA, called PFDA, with OpenMP. When parallelizing, one should aim for achieving a trade-off between improving performance, while minimizing the overheard of the parallelized mechanisms which includes communication, synchronization between threads, memory shar-



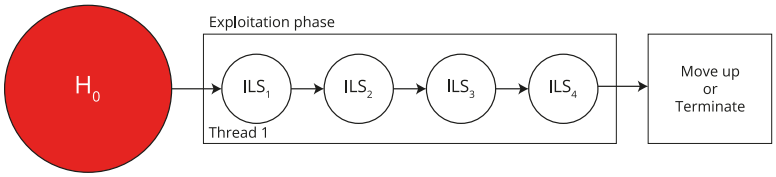
(a) First decomposition



(b) Evaluation of spheres (Exploration)



(c) Decomposition



(d) Exploitation phase

Figure 5: Illustration of Exploration phase (a) and (b) and the Exploitation phase (c) and (d) of a 2D problem with fractal depth 2 on a single threaded environment. The sphere in red having the highest quality at level 1 (a) and being decomposed (c) for exploitation phase (d).

ing and simplicity of implementation.

The idea behind parallelizing FDA was to remove the bottlenecks mentioned earlier, i.e. the exploration and exploitation phases. They are also the steps when function evaluations are consumed. Hence, these two phases need to be parallelized.

The Figure 6 illustrates the used strategy based on the previous example. Figures 6b and 6d represent the parallelized version of the exploration and exploitation phases, handling hyperspheres simultaneously, respectively.

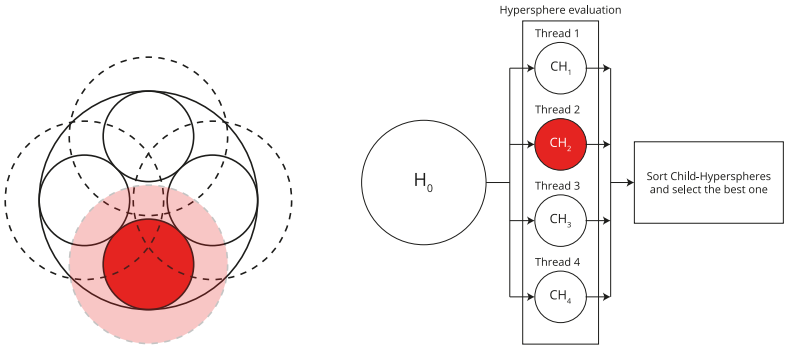
The initialization phase remains on a single thread, the hypersphere is being decomposed and only at this point the exploration phase starts. Instead of evaluating hyperspheres one at a time, from one to N hyperspheres with $N = 2 \times D$, PFDA is able to evaluate hyperspheres in parallel. The algorithm returns to a mono-threaded state and sort all hyperspheres, selecting the best one to be decomposed. This is being repeated until the last level k is reached. At this point the most promising region is decomposed triggering different instances of ILS. Then, $2 \times D$ generated hyperspheres are exploited in parallel. Once all hyperspheres have been exploited, PFDA terminates if stopping criterion is reached or backtracks in the search tree otherwise.

In other terms, PFDA alternates between mono-threaded and multi-threaded phases which corresponds to the well known $Fork/Join$ model. It is important to notice that the algorithm was designed to be easy to implement.

Regarding the programming environment for implementation, OpenMP is commonly used in the literature for multi-threaded environments, and stands out in terms of popularity, performance and simplicity of implementation [29, 30, 31].

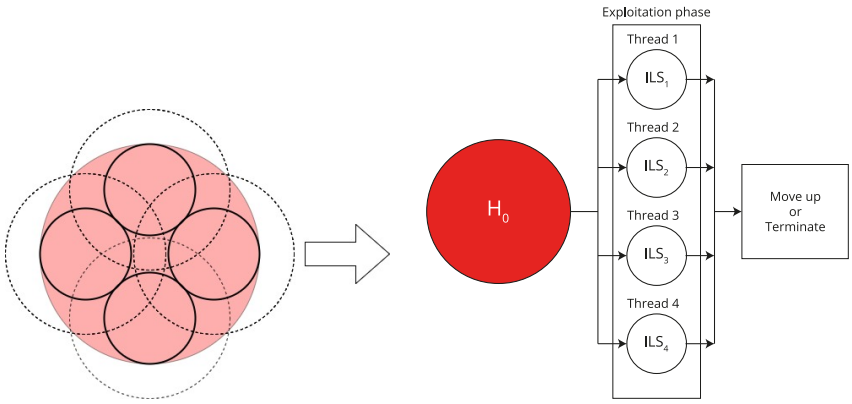
6. Results and Discussions

In this section, the obtained results are presented and analyzed. When adapting an existing metaheuristic it is important to be able to measure the benefits of the improvement. In this case, the main concern being the computa-



(a) First decomposition

(b) Evaluation of spheres (Exploration)



(c) Decomposition

(d) Exploitation phase

Figure 6: Illustration of the proposed strategy. Illustration of Exploration phase (a) and (b) and the Exploitation phase (c) and (d) of a 2D problem with fractal depth 2 on a multi-threaded environment. The child-hypersphere in red having the highest quality at level 1 (a) is being decomposed (c) for exploitation phase (d) which is also ran on a multi-threaded environment

tional time of the algorithm, this study will focus on the SpeedUp criteria [32]. This metric is defined by:

$$S = \frac{T_1}{T_n} \tag{6}$$

where S represents the SpeedUp, T_1 the execution time of the algorithm on a single thread and T_n, the execution time on n threads.

As shown in [32] this is not a valid comparison for non-deterministic algorithms. Originally FDA is deterministic, however, parallelizing the exploration phase adds a stochastic effect. Therefore, the SpeedUp remains suited for evaluating our approach.

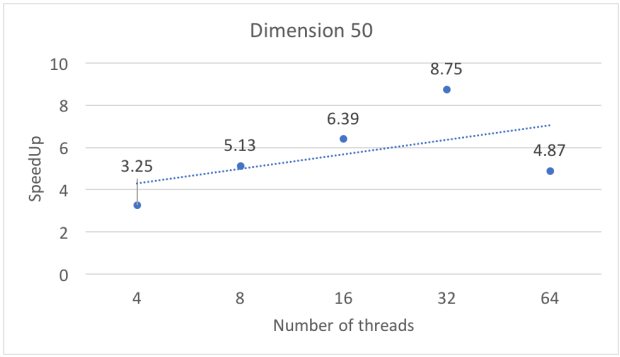
6.1. Performances evaluation

To evaluate the performance of the proposed algorithm on the large-scale continuous optimization benchmark of the Special Issue of Soft Computing on Scalability of Evolutionary Algorithms (SOCO 2011) was considered. This benchmark is composed of six functions from the CEC'2008 special session and competition on large-scale global optimization ([33]), and other problems generated by hybridizing these functions.

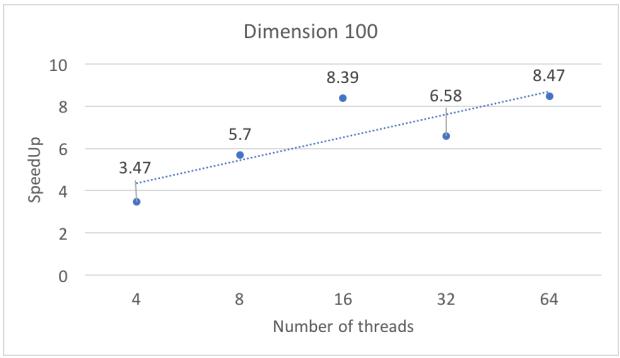
The comparison was performed between the computation time taken by PFDA and that of FDA to solve the benchmark. For the sake of the comparison, the stopping criterion of the benchmark was conserved: the number of functions evaluations set to $5000 \times D$, D *being the dimension of the problem.* In addition, only the dimensions $D = 50$, $D = 100$ and $D = 1000$ have been studied.

The machine used for experimentations has the following characteristics: a processor Intel Xeon E5-2686 v4 with 256GB of RAM with the technology Intel Turbo Boost Technology. The SpeedUp has been computed on the following number of threads: 4, 8, 16, 32, 64.

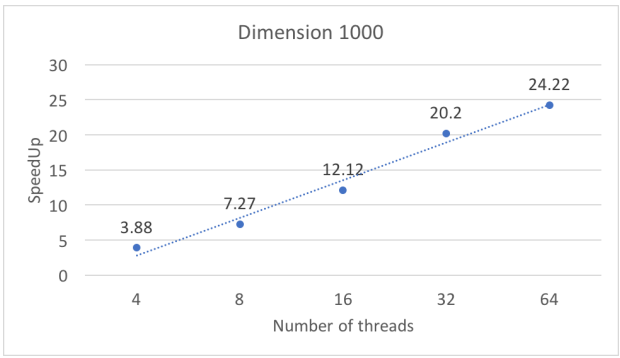
In Figure 7 variations of the SpeedUp over the number of threads are presented. One can see that the increase of the number of threads allows to reduce significantly the running time to reach the stopping criterion. It can also be noticed that for small dimensions, the increase of the number of threads does not automatically decreases the running time. However, for large problems, it is clear that the increase of the number of threads significantly decreases the execution time. The Figure 7c illustrates this remark in case of the dimension $D = 1000$, where the SpeedUp is equal to 24.22 with 32 threads.



(a)



(b)



(c)

Figure 7: SpeedUp versus the number of threads for solving the 19 functions (combined). (a) $D = 50$, (b) $D = 100$, (b) $D = 1000$.

To analyze the performances regarding different kind of problems, a focus was made on first six functions F_1-F_6 [33], on the dimension $D = 1000$. These six functions represent the different types of problems: separable and non-separable. The best obtained SpeedUp found is equal to 27.73 in case of a non-separable problem (Rosenbrock F_3 function). The different SpeedUps obtained for the previous considered problems are presented in Figure 8. It can be noticed that in all cases a linear tendency of the increase of the SpeedUp can be observed. This confirms the results illustrated in Figure 7c.

Regarding the quality of the final solution obtained by the algorithm, results of both versions (FDA and PFDA) are summarized in Table 1. It can be noticed from these results that when FDA found the optimum, PFDA also found it. However, the functions where FDA did not find the optimum, results of PFDA are far from the optimum. The quality of the solution, in this case, decreases with the increase of the number of threads. This is due to the stopping criterion being based on the number of function evaluations. Indeed, as $n = 64$, n hyperspheres are being explored in the same time, meaning that functions evaluations are performed in parallel. Hence, PFDA cannot exploit as deep the first hypersphere (supposed to be the most promising one) as FDA, which exploits them one at a time and can therefore go deeper in the first hypersphere. For instance, all functions evaluations are consumed in the first hypersphere generated on the last level k in case of the optimization of Rosenbrock problem via FDA.

Hence, FDA intensifies the search in the first hyperspheres more than PFDA can do. Indeed, PFDA exploits n hyperspheres at once.

It is obvious that the parallelized version needs more evaluations of the objective function to reach results similar to those of the single threaded version. In Figure 9, one can see the different SpeedUps obtained by FDA on single thread and PFDA on 64 threads. To analyze the performance in terms of SpeedUp when a target value of the objective function is considered as a stopping criterion. The Figure 10 presents obtained results. As it was expected, PFDA reaches similar results in a shorter computational time.

Table 1: Results error of the 19 functions of SOCO 2011 for FDA and PFDA.

Function	Original FDA	NB Thread 4	NB Thread 8	NB Thread 16	NB Thread 32	NB Thread 64
$F1$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$0.00E+00$
$F2$	$3.11E-01$	$3.67E+01$	$5.43E+01$	$6.43E+01$	$7.19E+01$	$8.51E+01$
$F3$	$1.13E+03$	$1.41E+03$	$2.41E+03$	$3.24E+03$	$4.46E+03$	$4.63E+03$
$F4$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$0.00E+00$
$F5$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$0.00E+00$
$F6$	$1.91E-12$	$1.92E-12$	$1.92E-12$	$1.89E-12$	$1.92E-12$	$1.89E-12$
$F7$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$0.00E+00$
$F8$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$0.00E+00$
$F9$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$0.00E+00$
$F10$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$0.00E+00$
$F11$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$0.00E+00$
$F12$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$1.45E-14$
$F13$	$7.69E+02$	$9.78E+02$	$1.05E+03$	$1.08E+03$	$1.18E+03$	$1.55E+03$
$F14$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$1.45E-07$
$F15$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$0.00E+00$
$F16$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$1.26E-07$
$F17$	$1.95E+02$	$3.76E+02$	$3.92E+02$	$4.30E+02$	$4.52E+02$	$6.88E+02$
$F18$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$5.89E-08$
$F19$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$0.00E+0$

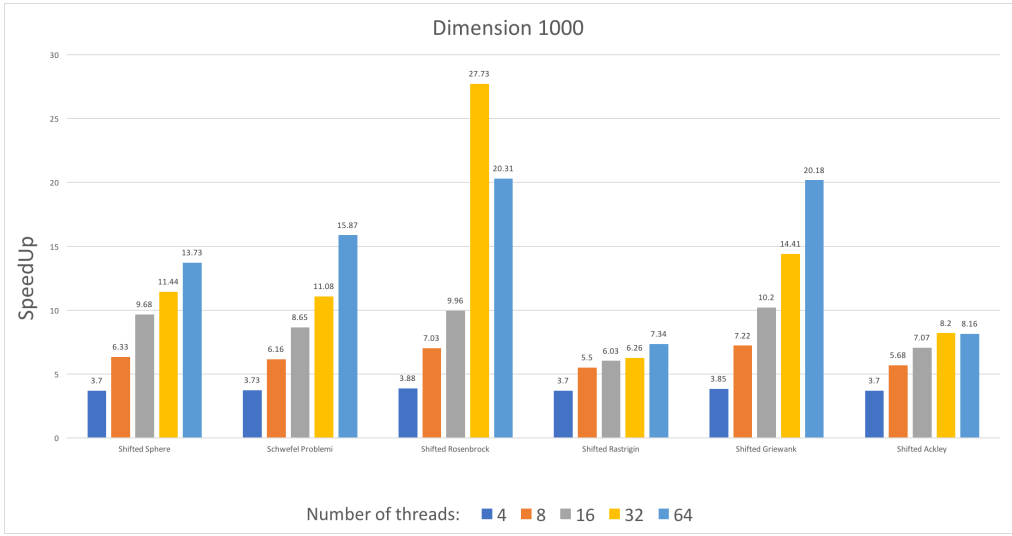


Figure 8: SpeedUp versus the number of threads concerning the six first functions of the SOCO 2011 benchmark.

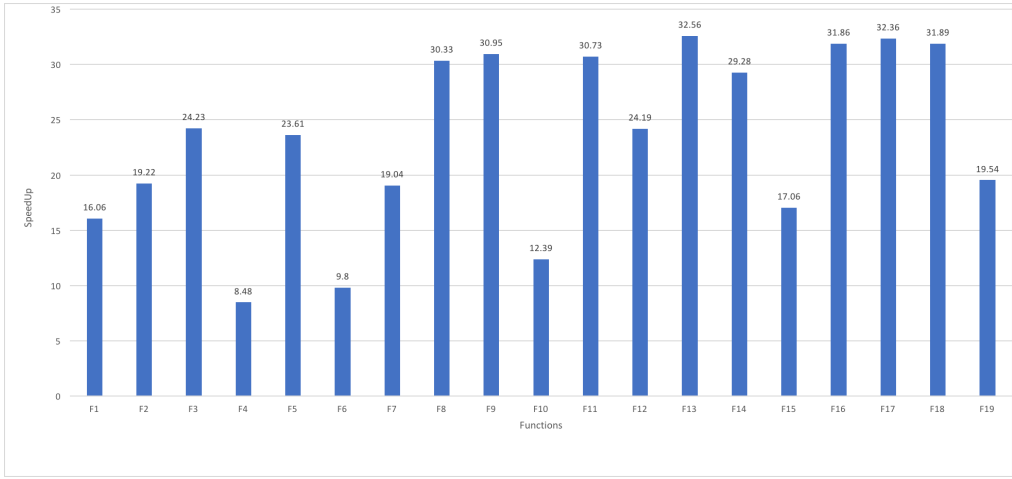


Figure 9: Obtained SpeedUps on a 64-thread environment with stopping criterion at $20000 \times D$ for both FDA and PFDA

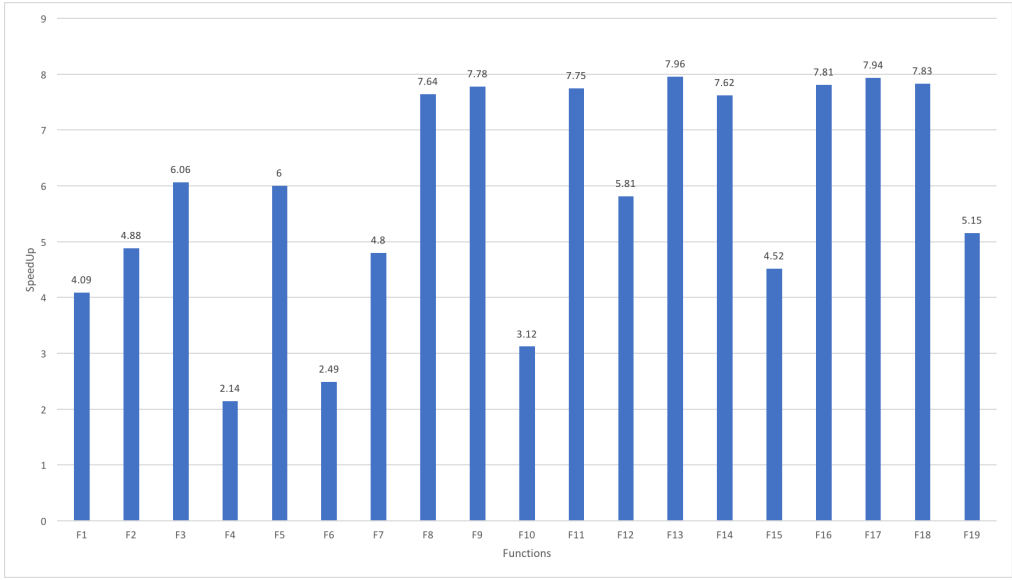


Figure 10: Obtained SpeedUps of single threaded FDA and 64-thread PDFA when a target value of the objective function is used as a stopping criterion.

6.2. Exploring higher dimension

In these experimentations the goal is to solve big optimization problems via PFDA. The considered problems are the first six functions of SOCO 2011 benchmark, where the dimension is $D = 5000$. The number of thread considered was 64. For the purpose of this study the stopping criterion will remain at $5000 * D$.

Originally, the benchmark SOCO 2011 sets the maximum dimension at $D = 1000$. To increase the dimension, instead of shifting the functions as provided by the benchmark, we shifted them randomly between in the interval $[L/10, U/10]$, where L is the lower-bound and U is the upper-bound.

The Figure 11 shows the obtained SpeedUps. Overall SpeedUps are higher than in the case of the dimension $D = 1000$ (Figure 8), except for the function F_1 Shifted Sphere. This can be explained by the nature of this problem (separable without local optima), both algorithms converges quickly to the optimum (it is known here). Therefore ILS is lost in trying to explore intensively, hyperspheres where an optimal or near optimal solution was already found. It is important to mention that this happens only if the stopping criterion is based on evaluation numbers. If PFDA is configured to stop when a target solution is found, then, the SpeedUp would be significantly higher.

7. Conclusion

In this work, a parallel version of the FDA algorithm, called PFDA, was proposed. The algorithm has been extensively tested on the SOCO 2011 Benchmark on large scale problems, going from 50 to 5000. Based on the SpeedUp criterion, it is easy to see that parallelizing FDA has improved significantly its performances on large-scale problems. However, during the exploration phase PFDA consumes lot of functions evaluations.

When the stopping criterion is a target value of the objective function a high SpeedUp was obtained. It can be concluded that this new approach enforces

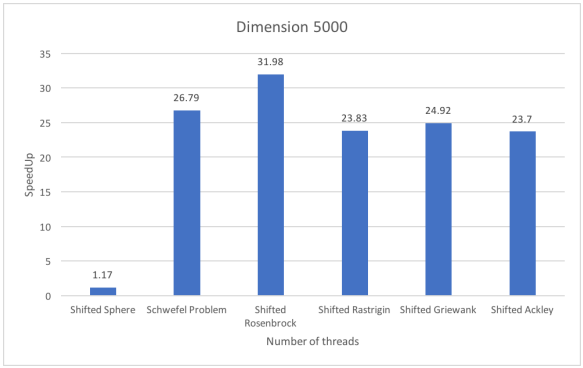


Figure 11: SpeedUp at dimension D = 5000 for the first six functions of the benchmark SOCO 2011 with number of threads $n = 64$

the original strengths as it converges significantly faster. However, PFDA remains less efficient in the case of highly non-separable problems. This is due to the heuristic used to explore hyperspheres, ILS. In work under progress, we aim to develop a new method for the exploration phase to enhance the performance to solve highly non-separable problems. In addition, the parallelization could go even further in running the exploration and exploitation phases on a heterogeneous environment with multi-nodes and/or GPU.

References

[1] F. Darema, S. Kirkpatrick, V. A. Norton, Parallel algorithms for chip placement by simulated annealing, IBM Journal of Research and Development 31 (3) (1987) 391–402.

[2] S. A. Kravitz, R. A. Rutenbar, Placement by simulated annealing on a multiprocessor, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 6 (4) (1987) 534–549.

[3] A. Casotto, F. Romeo, A. Sangiovanni-Vincentelli, A parallel simulated annealing algorithm for the placement of macro-cells, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 6 (5) (1987) 838–847.

[4] A. Delvacq, P. Delisle, M. Gravel, M. Krajecki, Parallel ant colony optimization on graphics processing units, Journal of Parallel and Distributed Computing 73 (1) (2013) 52 – 61.

[5] S. Gulcu, H. Kodaz, A novel parallel multi-swarm algorithm based on comprehensive learning particle swarm optimization, Engineering Applications of Artificial Intelligence 45 (2015) 33 – 45.

[6] E. Alba, G. Luque, S. Nesmachnow, Parallel metaheuristics: Recent advances and new trends 20 (2013) 1–48.

[7] T.-T. Vu, B. Derbel, Parallel branch-and-bound in multi-core multi-cpu multi-gpu heterogeneous environments, Future Generation Computer Systems 56 (2016) 95 – 109.

[8] 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 (Supplement C) (2017) 468 – 485.

[9] A. Reyes-Amaro, É. Monfroy, F. Richoux, Posl: A parallel-oriented metaheuristic-based solver language, in: Recent Developments in Metaheuristics, Springer, 2018, pp. 91–107.

[10] S. Santander-Jimnez, M. A. Vega-Rodrguez, Parallel multiobjective metaheuristics for inferring phylogenies on multicore clusters, IEEE Transactions on Parallel and Distributed Systems 26 (6) (2015) 1678–1692.

[11] X. Y. Zhang, J. Zhang, Y. J. Gong, Z. H. Zhan, W. N. Chen, Y. Li, Parallel genetic algorithm for the set cover problem and its application to large-scale wireless sensor networks, IEEE Transactions on Evolutionary Computation 20 (5) (2016) 695–710.

[12] M. Gorges-Schleuter, Asparagos an asynchronous parallel genetic optimization strategy, in: Proceedings of the Third International Conference on Genetic Algorithms, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1989, pp. 422–427.

[13] W. Baocheng, W. Tiane, W. Zenghui, The implementation of parallel ant colony optimization algorithm based on matlab, in: 2012 Third Global Congress on Intelligent Systems, 2012, pp. 27–29.

[14] H. Liu, Z. He, Parallel ant colony optimization algorithms for time series segmentation on a multi-core processor, in: 2012 4th International Conference on Intelligent Human-Machine Systems and Cybernetics, Vol. 1, 2012, pp. 340–343.

[15] P. Delisle, M. Krajecki, M. Gravel, Multi-colony parallel ant colony optimization on smp and multi-core computers, in: 2009 World Congress on Nature Biologically Inspired Computing (NaBIC), 2009, pp. 318–323.

[16] X. Situ, W. N. Chen, Y. J. Gong, Y. Lin, W.-J. Yu, Z. Yu, J. Zhang, A parallel ant colony system based on region decomposition for taxi-passenger matching, in: 2017 IEEE Congress on Evolutionary Computation (CEC), 2017, pp. 960–967.

[17] J. F. R. Herrera, J. M. G. Salmerón, E. M. T. Hendrix, R. Asenjo, L. G. Casado, On parallel branch and bound frameworks for global optimization, Journal of Global Optimization.

[18] D. R. Jones, C. D. Perttunan, B. E. Stuckman, Lipshitzian optimization without the Lipshitz coefficient, Journal of Optimization Theory Applications 79 (1) (1993) 157–181.

[19] M. Demirhan, L. Özdamar, L. Helvacğlu, c. I. Birbil, Fractop: A geometric partitioning metaheuristic for global optimization, J. of Global Optimization 14 (4) (1999) 415–436.

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

[21] J. He, M. Sosonkina, C. A. Shaffer, J. J. Tyson, L. T. Watson, J. W. Zwolak, Hierarchical parallel scheme for global parameter estimation in

systems biology, in: 18th International Parallel and Distributed Processing Symposium, 2004. Proceedings., 2004, pp. 42–.

[22] A. Al-Dujaili, S. Suresh, N. Sundararajan, Mso: a framework for bound-constrained black-box global optimization algorithms, Journal of Global Optimization (2016) 1–35.

[23] 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.

[24] 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.

[25] T. Liao, M. A. Montes de Oca, D. Aydin, T. Stützle, M. Dorigo, An incremental ant colony algorithm with local search for continuous optimization, in: Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation, GECCO '11, ACM, 2011, pp. 125–132.

[26] 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.

[27] O. R. Castro, R. Santana, J. A. Lozano, A. Pozo, Combining cma-es and moea/dd for many-objective optimization, in: 2017 IEEE Congress on Evolutionary Computation (CEC), 2017, pp. 1451–1458.

[28] 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), 2017, pp. 145–152.

[29] D. Akhmetova, R. Iakymchuk, O. Ekeberg, E. Laure, Performance study of multithreaded mpi and openmp tasking in a large scientific code, in:

2017 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), 2017, pp. 756–765.

[30] M. Arnautovi, M. Curi, E. Dolami, N. Nosovi, Parallelization of the ant colony optimization for the shortest path problem using openmp and cuda, in: 2013 36th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2013, pp. 1273–1277.

[31] D. D. Domenico, J. V. F. Lima, A. S. Charao, Openmp with parallel loops or asynchronous tasks: a performance evaluation focusing the nqueens benchmark, IEEE Latin America Transactions 15 (9) (2017) 1793–1800.

[32] E. Alba, G. Luque, Evaluation of parallel metaheuristics, 2006.

[33] K. Tang, X. Yáo, P. N. Suganthan, C. MacNish, Y.-P. Chen, C.-M. Chen, Z. Yang, Benchmark functions for the cec'2008 special session and competition on large scale global optimization, Nature Inspired Computation and Applications Laboratory, USTC, China (2007) 153–177.

Annexe A
Recall of original FDA performances

The benchmark SOCO 2011 has been used to test the performance of FDA, and in [8] authors compared the performances compared to DIRECT algorithm [18]. We recall that DIRECT is one of the most popular algorithm in the D&C based optimization approach. Its is known that DIRECT does not perform well on high dimensions. For this reason the comparison was performed on six functions with dimensions $D = 50$ and $D = 100$. For all functions in both dimensions, FDA performed significantly better than DIRECT. For instance, FDA found the global optimum in six problems out of twelve, while DIRECT found none.

Another comparison was performed with seven of the best SOCO 2011 participants. Among which can be found MA-SSW-Chains [23] or Multi-population Differential Evolution with balanced ensemble of mutation strategies for large-scale optimization (mDE-bES) [24]. To confirm the performance of FDA, a Friedman Rank Sum score was first computed to highlight the fact that FDA was ranked first. In addition a Wilcoxon pairwise test was used with an Holm procedure to adjust the p-values. This allowed to statistically show that FDA outperformed the other algorithms.

Finally, using the same functions and statistical tools, the third comparison has been conducted with five other recent metaheuristics, such as $\text{IACO}_{\mathbb{R}}$-Hybrid [25] or $2S - Ensemble$ [26]. Once again the competitiveness of FDA has been proven with regards to other algorithms.

All results and comparisons details can be read in the original paper [8].