Particle Swarm Optimization and Differential Evolution Methods Hybridized with Pattern Search for Solving Optimization Problems

Viviane J. Galvão[1], Helio J. C. Barbosa[1,2], and Heder S. Bernardino[2]

[1] Laboratório Nacional de Computação Científica (LNCC)
{vjgalvao, hcbm}@lncc.br
[2] Universidade Federal de Juiz de Fora (UFJF)
heder@ice.ufjf.br

Abstract. Derivative-free methods are being explored recently due to the increased complexity of the models used in the optimization problems, and the impossibility/inconvenience of using derivatives in several situations. However, those methods show some limitations due to their low convergence rate, and when the problem is high-dimensional. Metaheuristics are another commonly adopted type of search technique. Despite their robustness, metaheuristics require a large number of objective function evaluations to find an accurate solution. The combination of derivative-free optimization methods with population-based metaheuristics is analyzed here. Specifically, Particle Swarm Optimization and Differential Evolution are hybridized with Pattern Search technique. Also, an improvement of the conventional pattern search is proposed. Finally, computational experiments are performed to comparatively analyze the hybrid methods and the proposed pattern search.

Keywords: Derivative-Free Optimization, Pattern Search, Particle Swarm Optimization, Differential Evolution

1 Introduction

In Engineering, Economics, and Sciences, bound-constrained optimization problems with continuous variables arise where the use of derivatives is often avoided due to imprecision in their calculation or due to lack of access, as in the so-called black-box problems, where a simulator is required to compute the objective function and/or to check the constraints. Furthermore, such problems may require evaluations with high computational cost. These features make the use of numerical approximations for the derivatives, which employ finite difference, unattractive due to the high computational cost required, besides the potential noise in the objective function or constraints of the problem. These difficulties led to the use of an appropriate class of methods: the Derivative-Free Optimization (DFO) methods.

Since the 60s, the demand of problems motivated the creation of DFO methods which, until the 90s, did not have much theory associated with convergence rates. Nowadays, as seen in [14], the emphasis is in understanding the existing methods, with interest in their global convergence. The interest in these methods has grown as they are relatively easy to implement and to parallelize, and have wide applicability. They can be applied in engineering design optimization problems where each objective function evaluation can require minutes or days of CPU time. One can find a dense study of these methods in [5], including convergence analysis and some practical applications.

Among the DFO methods, one can cite the Direct Search methods which utilize only the objective function values in order to achieve convergence to a local optimum. More details regarding the Direct Search methods can be found in [11], such as a historical summary, new techniques, and constraint handling approaches. A Pattern Search method is also described by [11] and it consists in a technique that operates with exploratory moves, besides having a flexible structure that allows for the combination with other heuristics.

Although providing a wide applicability, the DFO methods have relatively slow convergence and do not achieve good results for high dimensions, when compared to derivative-based methods. One can expect success for DFO methods in problems with no more than a hundred variables and (i) which have a function evaluation with a high processing cost and (ii) where a high convergence rate is not the first goal. Some works try to improve the efficiency and the robustness of DFO methods. For instance, a simplex gradient is applied with pattern search methods in [6], and a particle swarm optimization technique is combined with a pattern search in [16].

On the other hand, one has the metaheuristics; they are popular due to their robustness, but in general require a high computational cost to obtain accurate results. Among them, one can cite the particle swarm optimization (PSO) [10], which is a population-based method inspired by the collective experience, and differential evolution (DE) [15], in which the movement operators are based on differences between vectors. One way to improve the results of metaheuristics is combining them with other heuristics. This process is known as hybridization [4], in which approaches are combined aiming to generate more efficient methods. For instance, a populated-based metaheuristc can be combined with a local search method to obtain a hybrid procedure able to find promising areas in the search space and capable of finding good results with a smaller number of objective function evaluations.

The proposal of this work is to combine PSO with Pattern Search, and DE with Pattern Search. The main idea is to reach an improved performance when solving continuous bound-constrained optimization problems. Also, a modification in the pattern search is proposed here in order to decrease the number of objective function evaluations. Computational experiments with problems often used in the literature are conducted in order to analyze the relative performance of the proposed techniques.

2 Pattern Search

Pattern search methods are directional direct search methods which use exploratory moves and a set of directions with appropriate features to guide the exploration of the neighborhood of a point in the search space. This set of directions must include at least one direction of descent over the iterations of the pattern search. To satisfy this condition a positive spanning or a positive base can be used, as seen in [12].

As presented in [2], the general structure of a pattern search method considers two steps: the search step and the poll step. Its general scheme is presented in the Algorithm 1. In the search step, for an iteration k, an objective function f, and x_{k-1} the best point obtained in the iteration $k-1$, a finite set of points of the search space is evaluated in order to find a point x such that $f(x) < f(x_{k-1})$. In a positive case, $x_k \leftarrow x$ and a new iteration is executed. The procedure to execute this step is not specified: any external heuristic can be executed in this step. This feature makes the pattern search methods very flexible.

In case of no success in the search step, the poll step is started, where a series of exploratory moves about the current best point x_{k-1} will be conducted. Those moves can be seen as a sampling of points in the neighborhood of x_{k-1}, in order to improve it. A set G of appropriate directions, a step-size α_{k-1}, and a set $P_k = \{x_{k-1} + \alpha d\}$ such that $d \in G$ are considered. Thus, if there is a point $x \in P_k$ such that $f(x) < f(x_{k-1})$, then $x_k \leftarrow x$; otherwise, $x_k \leftarrow x_{k-1}$.

In the end, the iteration can be classified as successful or unsuccessful. If x is found such that $f(x) < f(x_{k-1})$ in the search step or in the poll step, the iteration will be successful, and $\alpha_k = \gamma\alpha_{k-1}$, for $\gamma \geq 1$. If there is no improvement of the current best point in both steps, then the iteration will be unsuccessful and $\alpha_k = \beta\alpha_{k-1}$, for $\beta < 1$.

An application of the pattern search method to a class of molecular geometry problems can be found in [1]. In [6] the use of simplex derivatives is proposed to increase the efficiency of the pattern search methods. Also considering the pattern search methods flexibility, a combination of the Particle Swarm Optimization (PSO) meta-heuristic with a pattern search method is proposed in [16].

3 Adapted Pattern Search

As seen in the Algorithm 1, the pattern search has two steps: the search step and the poll step. In the poll step, a set of directions with appropriate features is used, more specifically positive spanning or a positive base, to execute exploratory moves in the neighborhood of a point in an attempt to find a direction of descent. When running such exploratory moves, it is required to evaluate the objective function in the points of the neighborhood obtained via the set of directions, in order to compare the quality of the current point and the neighborhood points.

The search of a direct of descent can be made by opportunistic or non-opportunistic way. In the opportunistic approach, when a direct of descent

Algorithm 1: Generalized Pattern Search

```
Data: $\boldsymbol{x}_0$, $\alpha_0$, $\gamma \geq 1$, $0 < \beta < 1$. Let $G$ a set of directions.
1  Let $k \leftarrow 0$;
2  while stop criteria false do
3      $k \leftarrow k + 1$;
4      $\boldsymbol{x}_k \leftarrow$ Search Step($\boldsymbol{x}_{k-1}$);
5      if $\boldsymbol{x}_k = \boldsymbol{x}_{k-1}$ then
6          $P_k \leftarrow$ Build the Neighborhood($\boldsymbol{x}_{k-1}$, $\alpha_{k-1}$, $G$);
7          for $\boldsymbol{y} \in P_k$ do
8              if $f(\boldsymbol{y}) < f(\boldsymbol{x}_{k-1})$ then
9                  $\boldsymbol{x}_k \leftarrow \boldsymbol{y}$;
10                 $\alpha_k \leftarrow \gamma\alpha_{k-1}$;
11             else
12                 $\alpha_k \leftarrow \beta\alpha_{k-1}$;
13             end if
14         end for
15     else
16         $\alpha_k \leftarrow \gamma\alpha_{k-1}$;
17     end if
18 end while
```

is found, the evaluation process of the neighborhood is finalized. The non-opportunistic all points of the neighborhood is evaluated, which can be computationally expensive.

As seen in [5], the maximal cardinality of a positive base is $2n$, where n is the problem size. In this way, when the cardinality of the set of directions is maximal and all directions are evaluated, we have the worst case scenario in terms of iteration cost. It is noted that even in the opportunistic approach, if there was no success in any direction or even if there was success in the last evaluated direction, each access to the poll step can lead to $2n$ function evaluations. Thus, there may be waste of computational resources.

To decrease the number of objective function evaluations in the Pattern Search, the last coordinate in which success was obtained is memorized and, in the next execution of the poll step, the search begins with the next coordinate. This is expected to reduce the occurrence of the worst case mentioned above in the neighborhood exploration process with opportunistic approach.

The procedure proposed for the adapted pattern search is illustrated in the Figure 1. As illustrated, in some iteration k the process of poll step starts evaluating the direction d_1, in green, and then the next ones will be evaluated until a direction of descent d_{n-2}, in pink, is found. Thus, the direction d_{n-2} is memorized and in the next iteration $k + 1$ of the poll step, the first direction to be evaluated will be d_{n-1} instead d_1, as is made in conventional pattern search.

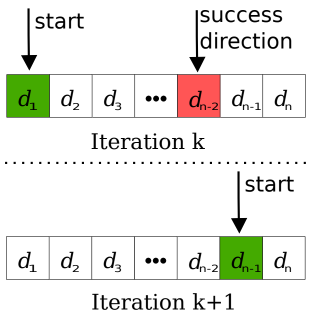


Fig. 1. Procedure of adapted pattern search

4 Differential Evolution

Differential Evolution (DE) [15] is a simple stochastic population-based algorithm which emerged as a very competitive evolutionary algorithm for continuous optimization [7]. Its basic proposal is to evolve a population where each individual is subject to mutation, crossover, and selection. In the DE, each individual $\boldsymbol{x}_i$ in the population is a candidate solution, represented by a position in the search space, $\boldsymbol{x}_i = [x_{i,1}, x_{i,2}, \ldots, x_{i,n}]$, where $i \in \{1, \ldots, N\}$, N is the population size, and n is the problem dimension.

DE's mutation process creates perturbations given by weighted differences between two individuals in the current population. In its simplest form, a mutated individual $\boldsymbol{v}_i$ is generated by:

$$\boldsymbol{v}_i = \boldsymbol{x}_{r_1} + F(\boldsymbol{x}_{r_2} - \boldsymbol{x}_{r_3}). \tag{1}$$

where $\boldsymbol{v}_i$ is the mutated individual, $r_1 \neq r_2 \neq r_3 \neq i$ are indexes of individuals randomly selected in $\{1, \ldots, N\}$, F is the weighting factor, and $i \in \{1, \ldots, n\}$.

The weighting factor in the Equation 1 is the parameter which controls the the size of the perturbation defined by the difference vector $(\boldsymbol{x}_{r_2} - \boldsymbol{x}_{r_3})$. Thus, the weighting factor should be small enough to allow for exploitation but large enough to maintain the diversity of the population and the global exploration. It is common to use $F \in [0.4, 1]$, as seen in [7].

The crossover process aims at increasing the diversity of the mutated individuals. The two most often used kinds of crossover methods are: the exponential crossover and the binomial crossover. In this work we adopted the binomial crossover. Thus, new individuals, the trial individuals, are created such that:

$$\boldsymbol{u}_i = \begin{cases} \boldsymbol{v}_i & \text{if } r_j \leq \text{CR} \\ \boldsymbol{x}_i & \text{if } r_j > \text{CR} \end{cases} \tag{2}$$

where $\boldsymbol{u}_i$ is a trial individual, $\boldsymbol{v}_i$ is a mutated individual, $\boldsymbol{x}_i$ is an individual of the current population, r_j is a number drawn from a uniform distribution in $(0, 1)$, CR is a parameter defined in $[0, 1]$, and $i \in \{1, \ldots, n\}$.

The parameter CR is the crossover rate and represents the probability of the new individual to be more similar to the individual of the current population or with the mutated individual. If the value of CR is close to 1 the trial individual will be more similar to the mutated individual.

Finally, to construct the new population for the next generation, the selection process preserves the best individuals between the current ones and the trial individuals by means of a greedy criterion given by:

$$\boldsymbol{x}_i^{t+1} = \begin{cases} \boldsymbol{u}_i^t & \text{if } f(\boldsymbol{u}_i^t) \leq f(\boldsymbol{x}_i^t) \\ \boldsymbol{x}_i^t & \text{otherwise} \end{cases} \tag{3}$$

After the presentation of the main operations of the Differential Evolution method, its basic structure is described in the Algorithm 2.

Algorithm 2: Differential Evolution

```
Data: F, CR, N
Initialize the individuals population $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N$;
Let $\hat{x}$ the best individual of the population;
while stop criteria false do
    for $i = 1, \ldots, N$ do
        $\boldsymbol{v}_i \leftarrow$ Mutation($\boldsymbol{x}_i$, F);
        $\boldsymbol{u}_i \leftarrow$ Crossover($\boldsymbol{v}_i$, $\boldsymbol{x}_i$, CR);
    end for
    for $i = 1, \ldots, N$ do
        if $f(\boldsymbol{u}_i) < f(\boldsymbol{x}_i)$ then
            $\boldsymbol{x}_i \leftarrow \boldsymbol{u}_i$;
            if $f(\boldsymbol{x}_i) < f(\hat{x})$ then
                $\hat{x} \leftarrow \boldsymbol{x}_i$;
            end if
        end if
    end for
end while
return $\hat{x}$;
```

Actually, there are different kinds of DE, depending on the mutation and the crossover approaches utilized. The simplest DE form is referred to in the literature as DE/rand/1/bin and uses the mutation in Equation 1 and the binomial crossover in Equation 2. The terminology usually adopted is DE/x/y/z, where x represents the base vector to be perturbed in the mutation, y is the number of difference vectors of perturbation in the mutation, and z denotes the type of crossover used, usually the exponential crossover (denoted by "exp") or the binomial crossover ("bin"). The DE variants used in this work are the DE/rand/1/bin, DE/best/1/bin, and DE/target-to-best/1/bin, all with a binomial crossover, and defined in Equations 1, 4, and 5, respectively.

$$\text{(DE/best/1/bin)} \quad v_i^t = x_{best}^t + F(x_{i,r_1}^t - x_{i,r_2}^t) \tag{4}$$

$$\text{(DE/target-to-best/1/bin)} \quad v_i^t = x_i^t + F(x_{best}^t - x_i^t) + F(x_{i,r_1}^t - x_{i,r_2}^t) \tag{5}$$

When the problem has bound-constraints, those are enforced by simply projecting the violated component into the feasible range: the closest bound value is assigned to the violated design variable value.

5 Particle Swarm Optimization

The particle swarm optimization (PSO) [10], is a stochastic population-based metaheuristic which handles continuous problems and is inspired by collective intelligence mechanisms and by the social behavior of groups of animals, such as birds and fishes. Its basic idea is to graphically simulate the choreography of the group, and the sharing of information and knowledge in the food search process.

In this model, the birds are considered as particles and each represents a candidate solution in the search space. Thus, each particle of the population is associated with a triple $(\boldsymbol{x}, \boldsymbol{\nu}, \boldsymbol{x}^*)$, which corresponds to the current position of the particle, its velocity, and the best position visited by the particle, respectively.

Denoting the population size by N, while some stop criterion is not achieved, the velocity (ν) and the position (x) of each particle are updated by:

$$\nu_i = \delta\nu_i + \eta\omega_{1i}(x_i^* - x_i) + \psi\omega_{2i}(\hat{x}_i - x_i) \tag{6}$$

$$x_i = x_i + \nu_i \tag{7}$$

where $\hat{x}$ is the best global particle, δ is the inertia factor of the particle, $\eta > 0$ is a learning factor, $\psi > 0$ is a social factor, ω_{1i} and ω_{2i} are random numbers uniformly distributed in $(0, 1)$.

The basic structure of the PSO is described in the Algorithm 3. For each particle, one checks if the new position is better than the current one; if so, the best point is updated, and it is compared with the best global point. The iterations have success when the best global point is improved.

A technique to calculate the inertia factor is to initialize it with a relatively high value and then decrease it gradually to a smaller value. This because the inertia factor can be seen as an environment fluidity of the medium where the particles move. For high fluidity values the environment is less viscous, thus allowing for more exploration. For low values, the environment viscosity is high and the search is more focused, thus increasing exploitation.

Although both metaheuristics calculate differences in their movement operators, as seen in equations 1 and 6, one can notice that DE uses differences between vectors while PSO subtracts vector components independently.

When the problem has bound-constraints, those are often handled by projection, as explained for the DE case.

6 The Proposed Hybrid Techniques

The hybrid techniques proposed here consist in the combination of Differential Evolution with Pattern Search and of Particle Swarm Optimization with Pattern Search. Also, the metaheuristics are combined with the adapted Pattern Search.

Algorithm 3: Particle Swarm Optimization

```
Data: $\delta$, $\eta$, $\psi$, N
1  Initialize the particle population $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$;
2  Initialize the velocities $\boldsymbol{\nu}_1, \ldots, \boldsymbol{\nu}_N$;
3  Let $\hat{x}$ the best global position;
4  while stop criteria false do
5      for $i = 1, \ldots, N$ do
6          for $j = 1, \ldots, D$ do
7              $\nu_{ij} = \delta\nu_{ij} + \eta\omega_{1j}(x^*_{ij} - x_{ij}) + \psi\omega_{2j}(\hat{x_{ij}} - x_{ij})$;
8              $x_{ij} = x_{ij} + \nu_{ij}$;
9          end for
10         if $f(\boldsymbol{x}_j) < f(\boldsymbol{x}^*_j)$ then
11             $\boldsymbol{x}^*_j \leftarrow \boldsymbol{x}_j$;
12             if $f(\boldsymbol{x}_j) < f(\hat{x})$ then
13                 $\hat{x} \leftarrow \boldsymbol{x}_j$;
14             end if
15         end if
16     end for
17 end while
```

The general structure of the hybrid DE method consists, for each iteration, in executing the DE algorithm and verifying if some individual is better than the current best individual. In that is the case, the best individual is updated, the poll step of the pattern search is not executed and the iteration is successful; otherwise, the poll step of pattern search is performed on the current best individual of the population, and the iteration is unsuccessful.

In the same way, the general structure of the hybrid PSO method consists in executing the PSO algorithm and verifying if some particle is better than the current best particle. If the iteration is successful, the best particle is updated and the poll step of the pattern search is not executed; otherwise, the poll step of the pattern search is performed on the current best particle.

Figure 2 presents a flowchart shared by both (DE and PSO) proposed hybrids. The central block contains the particular operational details of each metaheuristic, where for the hybrid DE the individuals will be subject to mutation, crossover, and selection operators, and for the hybrid PSO the particles will have their velocities and positions updated.

7 Computational Experiments

Computational experiments were conducted to comparatively evaluate the performance of the proposed techniques. We used a benchmark composed by 100 bound-constrained optimization problems implemented in AMPL [9], which were obtained from a collection of problems from [16]. For the comparisons of the obtained results, Performance Profiles [8] were used, with the average objective function value as the performance measure.

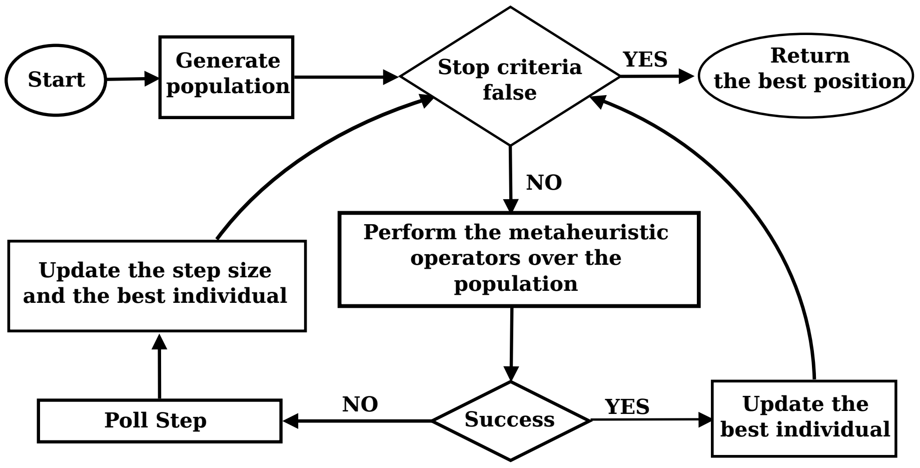

Start
Generate population
Stop criteria false
YES
Return the best position
NO
Perform the metaheuristic operators over the population
Update the step size and the best individual
Poll Step
NO
Success
YES
Update the best individual

Fig. 2. Flowchart of the proposed hybridization of the metaheuristics and Pattern Search.

The maximum number of objective function evaluations was equal to 2000 and 30 independent runs were performed for each technique in each test-problem. The results found by the proposed approaches are compared to those obtained by PSwarm [16], an implementation of PSO with a Pattern Search.

The proposed techniques and their respective labels adopted are as follows: (i) PSO is the Particle Swarm Optimization method, (ii) PSO+PS is PSO with the Pattern Search, (iii) DE1, DE2 and DE3 are the corresponding DE variants DE/best/1/bin, DE/rand/1/bin and DE/target-to-best/1/bin, (iv) DE1+PS, DE2+PS, and DE3+PS are the DE methods with the Pattern Search. The label PSWARM makes reference to PSwarm of [16]; the PSOPS*, DE1PS*, DE2PS*, DE3PS* and PSWARM* have the adapted Pattern Search.

The parameters used in the performed experiments were obtained from the automatic parameter tuning tool `irace` [13], using the problems from [16] and allowing 2000 function evaluations. The standard PSO used $\eta = 0.3815$, $\psi = 0.3727$ and the weight δ was calculated by $\delta(t) = \delta_i - (\delta_f)(t/t_{max})$, where $\delta_i = 0.8314$, $\delta_f = 0.3165$, t is the function evaluation counter and t_{max} is the function evaluations maximum amount allowed. For the the DE variants the parameters are shown in the Table 1. The Pattern Search parameters are the same as those defined in [16], with $\gamma = 2.0$ for the expansion factor and $\beta = 0.5$ for the decreasing factor.

7.1 Results

The results found in the computational experiments are shown in the Figures 3 and 4 presenting performance profiles of the most efficient techniques studied and proposed.

Variants	F	CR	N
`DE/rand/1`	0.66	0.79	10
`DE/best/1`	0.55	0.93	32
`DE/target-to-best`	0.80	0.83	11

Table 1. Parameters for the DE variants.

Initially, to detect the best performing techniques, a preliminary study of the results using the area under the curve $\rho(\tau)$, as in [3], was made.

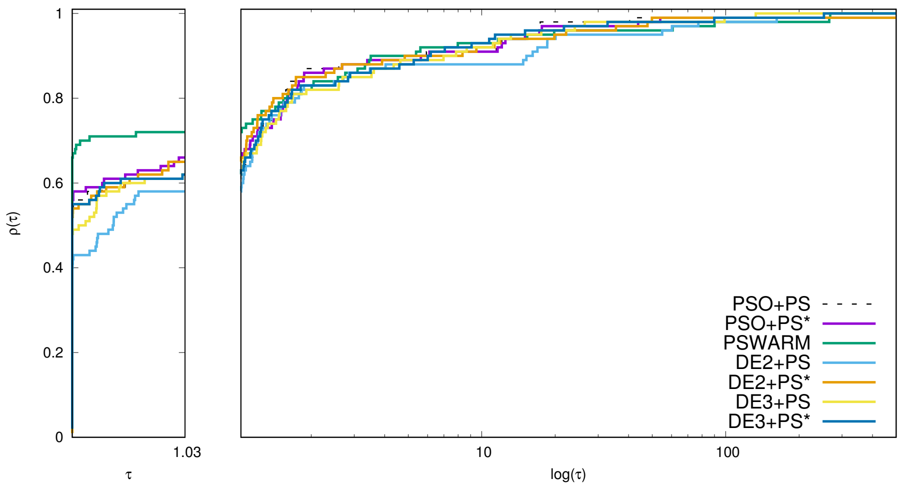
The Figure 3 shows the performance profile plots considering all problems. From this figure one can conclude that the PSWARM technique has the best performance (largest $\rho(1)$) with respect to the other techniques, obtaining the best solutions in 66% of the problems. Also, one can notice that the use of the adapted pattern search improves the performance of the PSO and the DE with respect to their variants with the original pattern search. Moreover, the metaheuristics with pattern search and adapted pattern search are more efficient than the original metaheuristics.

The Figure 4 shows the performance profile plots considering the problems with dimension greater than 10. This figure shows that PSWARM* obtained the best performance (largest $\rho(1)$), being the top performer in 43% of the problems. Moreover, PSWARM* is the most robust method, and its performance remains higher than the performances of the other methods. One can notice that the performance curve of PSWARM* does not appear in the Figure 3 because it has the third lowest area under the curve value and, hence, was classified as less efficient than the others for all problems, as seen in [3]. Thus, the proposed pattern search improved the performance and robustness of PSWARM for problems with dimension greater than 10, although for the remaining problems - with low dimensions - the proposed pattern search did not work very well.

8 Concluding Remarks and Future Works

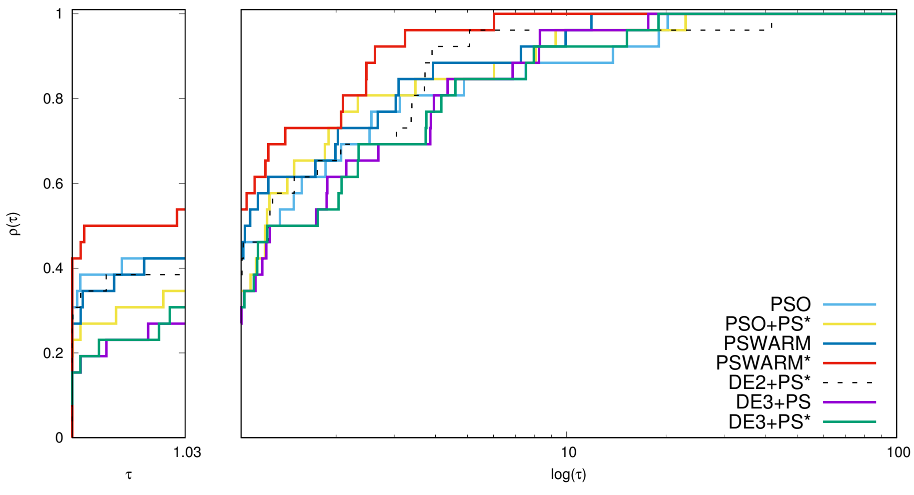
The hybridization of a Pattern Search with (i) Particle Swarm Optimization and (ii) Differential Evolution is proposed here. Also, an improvement of the original Pattern Search procedure is presented.

From the analysis of the results obtained in the computational experiments, it is possible to conclude that the performance of hybrid metaheuristics is very competitive with respect to the studied metaheuristics performance. Moreover the use of the adapted pattern search improves the performance of the metaheuristics with original pattern search, indicating that to avoid of many accumulated function evaluations at one stage of the search may result in good responses. Also, the PSwarm with the adapted pattern search proposed here improves the performance of the original PSwarm for problems with dimension larger than 10, but the original PSwarm leads to the best performance for all problems.



(a) (b)

Fig. 3. Performance profiles - All techniques, 100 problems. Figure (a) shows the performance profiles for $\tau \in [1, 1.03]$; (b) shows, in logarithmic scale, the performance profiles for $\tau \in [1.03, 500]$.



(a) (b)

Fig. 4. Performance profiles - All techniques, 26 problems with dimension greater than 10. Figure (a) shows the performance profiles for $\tau \in [1, 1.03]$; (b) shows, in logarithmic scale, the performance profiles for $\tau \in [1.03, 100]$.

As future work, we plan to continue the investigation of new combinations, studying both metaheuristcs and other derivative-free techniques, as well as their application to practical problems, such as those found in structural optimization.

Acknowledgements

The authors thank the support from CAPES, CNPq (310778/2013-1), FAPEMIG (APQ-03414-15), and PPGMC/UFJF, and the comments from a reviewer.

References

1. Alberto, P., Nogueira, F., Rocha, H., Vicente, L.N.: Pattern search methods for user-provided points: Application to molecular geometry problems. SIAM Journal on Optimization 14(4), 1216–1236 (2004)
2. Audet, C., Dennis Jr, J.E.: Analysis of generalized pattern searches. SIAM Journal on Optimization 13(3), 889–903 (2002)
3. Barbosa, H.J.C., Bernardino, H.S., Barreto, A.M.S.: Using performance profiles to analyze the results of the 2006 cec constrained optimization competition. In: IEEE Congress on Evolutionary Computation. pp. 1–8. IEEE (2010)
4. Blum, C., Roli, A.: Hybrid Metaheuristics: An Introduction, pp. 1–30. Springer Berlin Heidelberg, Berlin, Heidelberg (2008)
5. Conn, A.R., Scheinberg, K., Vicente, L.N.: Introduction to Derivative-Free Optimization. Society for Industrial and Applied Mathematics (2009)
6. Custódio, A.L.: Aplicações de Derivadas Simplécticas em Métodos de Procura Directa. Ph.D. thesis, Universidade Nova de Lisboa (2007)
7. Das, S., Suganthan, P.N.: Differential evolution: A survey of the state-of-the-art. Trans. Evol. Comp 15(1), 4–31 (Feb 2011)
8. Dolan, E.D., Moré, J.J.: Benchmarking optimization software with performance profiles. Mathematical Programming 91(2), 201–213 (2002)
9. Fourer, R., Gay, D.M., Kernighan, B.W.: A modeling language for mathematical programming. Management Science 36(5), 519–554 (1990)
10. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proc. of the IEEE International Conference on Neural Networks. vol. 4, pp. 1942–1948. IEEE (1995)
11. Kolda, T.G., Lewis, R.M., Torczon, V.: Optimization by direct search: New perspectives on some classical and modern methods. SIAM Review 45(3), 385–482 (2003)
12. Lewis, R.M., Torczon, V.: Rank ordering and positive bases in pattern search algorithms. Tech. rep., DTIC Document (1996)
13. López-Ibáñez, M., Dubois-Lacoste, J., Cáceres, L.P., Birattari, M., Stützle, T.: The irace package: Iterated racing for automatic algorithm configuration. Operations Research Perspectives 3, 43 – 58 (2016)
14. Rios, L.M., Shahinidis, N.V.: Derivative-free optimization: a review of algorithms and comparison of softwares implementations. Journal of Global Optimization 56(3), 1247–1293 (2013)
15. Storn, Rainer & Price, K.: Differential evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces, vol. 3. ICSI Berkeley (1995)
16. Vaz, A.I.F., Vicente, L.N.: A particle swarm pattern search method for bound constrained global optimization. J. of Global Optimization 39(2), 197–219 (2007)