On a Cooperative Hybrid Algorithm Based on Harmony Search and Differential Evolution for Numerical Optimization

Gustavo Nogueira de Sousa  
Programa de Pós-Graduação em Engenharia da Computação e Sistemas (PECS)  
Universidade Estadual do Maranhão (UEMA)  
São Luís, Maranhão, Brasil  
sougesta@gmail.com

Omar Andres Carmona Cortes  
Departamento de Computação (DComp)  
Instituto Federal de Educação, Ciência e Tecnologia do Maranhão (IFMA)  
São Luís, Maranhão, Brasil  
omar@ifma.edu.br

ABSTRACT

Hybrid algorithms aim to mix features from two or more evolutionary/swarm algorithms to improve both the exploration and exploitation abilities of the algorithm. Generally, hybrid algorithms improve the same quality of solution than the canonical ones, in the worst case scenario. However, it is common that hybrid algorithms present better outcomes than the canonical ones. In this context, this paper proposes a cooperative hybrid algorithm based on Harmony Search and Differential Evolution named HS-DE. The algorithm has been tested in five benchmark functions well known in the literature. Results have shown that HS-DE presents better solutions than Genetic Algorithms, Particle Swarm Optimization, Differential Evolution, and Harmony Search in all benchmark functions.

KEYWORDS

Hybrid Algorithm, Harmony Search, Differential Evolution, Numerical Optimization

1 INTRODUCTION

Numerical Optimization problems exist widely in different areas of scientific research and engineering practice [1]. It is a tool for solving practical unconstrained problems that can be devised by many variables. Its primary purpose is to discover the best values for design variables and objective functions that are not known precisely [2]. In general, unconstrained problems can be devised by test functions, also known as benchmark functions. Benchmarks are artificial problems that can be used to evaluate the behavior of an algorithm in diverse and challenging situations [3], such as, functions containing multiple local optima (multimodal), and hard quadratic functions, such as, the Rosenbrock [4] function.

Classical methods for solving numerical optimization are fast; however, they present two critical drawbacks. The first one is related to the number of variables, i.e., they can be used only in a small number of variables. The second one regards to the differentiability of the objective function, i.e., classical methods demand that functions are derivable, which is not common in real-world problems. Thus, meta-heuristics, such as Differential Evolution (DE) [5], Particle Swarm Optimization (PSO) [6], Genetic Algorithms (GA) [7], and Harmony Search [8], represent a viable set of techniques suitable for solving this unconstrained and sometimes non-differentiable functions.

Even though traditional meta-heuristics have proved to be efficient, hybrid algorithms tend to present better results than canonical meta-heuristics, as we can see in [9], [10], and [11]. In this context, this work presents a hybrid algorithm based on Harmony Search and Differential Evolution for solving Numerical Problems, which are similar to those found in engineering problems. We used five benchmark functions well known in the literature: Rosenbrock[4], Sphere [12], Schwefel[13], Rastrigin [14], and Griewank[15].

In this context, this work is divided as follows: Section 2 introduces the main concepts on numerical optimization. Section 3 shows the meta-heuristics used in this work, along with our hybrid algorithm proposal. Section 4 presents the results of our cooperative hybrid algorithm and compares it against canonical meta-heuristics. Finally, in Section 5, we draw some conclusions of this work.

2 NUMERICAL OPTIMIZATION

The unconstrained numerical optimization proposes to minimize or maximize an objective function depends on floating point variables, with no restrictions at all on the values of these variables [16]. Mathematically, it is min or max\( f(x) \), where \( x \in \mathbb{R}^n \) and \( n \geq 1 \). Thus, a solution \( x^+ \) is a global solution of a minimization problem if \( f(x^+) < f(x) \) \( \forall x \); analogously, it is a solution of a maximization problem if \( f(x^+) > f(x) \) \( \forall x \).

Regardless of the kind of optimization, if we want to use a GA for this kind of problem, it is mandatory that \( n > 1 \). The variable \( n \) regards to the dimensionality of the search space that is an essential factor in the problem complexity, since the higher the dimension, the higher the probability of getting trapped in a local optima [17].

Two other properties are crucial in numerical optimization: separability and multi-modality. The separability involves the possibility of dividing \( f(x) \) into two or more functions. Consequently, non-separable functions are more challenging to optimize than separable ones. Multi-modality concerns the existence of many local optima. Thus, non-separable and multi-modal functions represent a more significant challenge to solve than the other ones.

As previously mentioned, we will test our code using five unconstrained continuous numerical benchmarks functions: Rosenbrock, Sphere, Schwefel, Rastrigin, and Griewank. Table 1 presents the function, benchmarks properties (Separability, Modality, and Differentiability), the domain, and the global optima. The domain is a constraint for each gene, i.e., the lower and upper bounds. The optimal solution is the minimum value that the benchmark can reach. The separability represents if the function is separable, i.e., if the function can be split into two or more functions. In other words, a function of \( p \) variables is called separable, if it can be written as a
sum of $p$ functions of just one variable [18]. Finally, the modality regards to the existence of many local optima. In this context, non-separable and multi-modal functions are harder to solve than the other ones. Figures 1 to 5 show the appearance of each benchmark function on a three-dimensional space (two variables).

3 META-HEURISTICS

A meta-heuristic is a top-level strategy that guides an underlying heuristic solving a given problem [19]. In this context, this section presents two different meta-heuristics used in this work: Harmony Search and Differential Evolution. Then, we present our approach that cooperatively uses both meta-heuristic.

3.1 Harmony Search

Music harmony is a combination of sounds considered pleasing from an aesthetic point of view [8]. Harmony is a special relationship between several sounds that are pleasant to humans. In this context, composers aim to obtain the best combination of sound waves that are already in the musician’s memory. The process of choosing harmonies from their memory ends up being an optimization process. According to Yang [20], Harmony Search is a music-based optimization algorithm inspired by the searching for the perfect state of harmony.

The HS algorithm is presented in Algorithm 1, in which there are two basics steps: (i) improvising a new harmony and (ii) updating the harmony memory. The improvisation comes from the musician experience, and the updating is performed only if the improvisation sounds good.

As done in other meta-heuristics, the initialization process of the Harmony Memory is done randomly. Then the improvisations are performed by using the Equation 1, in which HMS is the harmony memory size, HMCR is the Harmony Memory Consideration Rate, between zero and one, of choosing one value from the historical
values stored in the Harmony Memory. Whereas (1-HMCR) is the rate of randomly selecting one value from the domain of each dimension, and HMS represents the harmony memory size. If we use, for instance, an HMCR equals to 95%, it means that variables \(x_i\) will be mostly chosen from the HM.

\[
x_i = \begin{cases} \text{if a random number is between 0 and 1} & \text{HMCR} \\
x_i \in X_i & 1 - \text{HMCR} \end{cases}
\]

The second step of the improvisation is to verify if the new harmony \(x'\) needs a pitch adjustment. The process is done on each variable according to Equation 2, in which PAR is the Pitch Adjusting Ratio, which is a random number between zero and one.

\[
x_i = \begin{cases} \text{Yes} & \text{PAR} \\
\text{No} & 1 - \text{PAR} \end{cases}
\]

The DE algorithm begins initializing a random population and evaluates it. Then, the mutation process chooses three random individuals creating the vector \(v\), which is also called a vector of differences, in which \(F\) is a constant determined by the programmer. Then, a new individual is created by using a gene from \(v\) if a random number is less than CR (Crossover Rate); otherwise, the gene comes from popij. Finally, if the new individual is better than that one in the current population, the new one replaces it.

The strategy presented in the Algorithm 3 is called DE/Rand/1 because all individuals are randomly selected, and only one vector of differences is created. However, if \(indiv_i\) is replaced by the best solution in the population, the name of strategy changes to DE/Best/1.

### 3.3 The Cooperative Hybrid HS-DE Algorithm

Our proposal is presented in Algorithm 4. The first step is similar to the canonical HS in which the Harmony Memory is initialized according to the Harmony Memory Size and dimension. When iterations start, a temporary population is created using the HS algorithm and the HM. The same HM is evolved using Differential Evolution. Then, the fitness of HS and DE is computed. Afterward, all solutions from HM, HM', and DE' are put together, and only the best solutions go to the HM that will be used again by HS and DE algorithms. The process is repeated up to the stop criterion.

Figure 6 presents a flowchart in which is more evident how the algorithm HS-DE works. As we can see, we have two independent flows when HS and DE update their populations. Then all populations (HM, HM', and DE') are getting together to choose the best solutions that undergo the next iteration. On the one hand, we
Algorithm 4 - HS-DE Pseudo Code

\begin{verbatim}
HM ← InitHarmonyMemory(HMS)
while stop Criterion not reached do
    HM' ← HarmonySearch(HM)
    DE' ← DifferentialEvolution(HM)
    Pop ← Join(HM, HM', DE')
    fit ← Evaluate(Pop)
    HM ← Best(fit, HSM)
end while
\end{verbatim}

perform more comparison than other algorithms; that is why we use the number of calls to objective function as a stop criterion. On the other hand, this approach is natural to execute in parallel, which is not the case in this paper.

\begin{table}[h]
\centering
\begin{tabular}{|l|l|}
\hline
Algorithm & Parameters used in the algorithms \\
\hline
GA & \(p_c = 0.8, p_m = 0.05\), with and without elitism \\
PSO & \(w = 0.7, c_1 = c_2 = 2.0, v_{\text{min}} = -100, v_{\text{max}} = 100\) \\
DE & \(F = 0.3, p_c = 0.6, \text{DE/Rand/1}, \text{and DE/Best/1}\) \\
HS & \(\text{PAR} = 0.3, \text{HMCR} = 0.95, \text{bandwidth} = 0.05\) \\
HS-DE & \(\text{PAR} = 0.3, \text{HMCR} = 0.95, \text{bandwidth} = 0.05, F = 0.3, p_c = 0.6, \text{DE/Rand/1}\) \\
\hline
\end{tabular}
\caption{Parameters used in the algorithms}
\end{table}

4 EXPERIMENTS

In this section, we present how the set up of the execution environment and the machine configuration. We also specify the parameters of the algorithms. Further, the results show a comparison between our approach and some other meta-heuristics such as PSO, GA, and the previously presented meta-heuristics, DE, and HS.

4.1 Setup

The experiments have been conducted in an i5 processor 8th generation with 8GB of RAM in a Linux Mint. The hybrid algorithm was compared against GA, PSO, and DE. Concerning GA, the algorithm was tested with and without elitism. Regarding DE, we used two strategies: DE/Rand/1 and DE/Best/1. The algorithms were executed in 50 trials, dimension equals 30, and population size equals 50. Table 2 presents all parameters we have used.

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{HS-DE_Flowchart.png}
\caption{HS-DE Flowchart}
\end{figure}

4.2 Results

Two sets of experiments have been conducted. To a fair comparison between algorithms, the stop criterion was the number of calls to the objective function. In doing so, the first set of experiments uses 25000 calls, and the second one uses 50000 calls. GA-E represents GA using elitism. DE1 is the DE/Rand/1 strategy, and DE2 is the DE/Best/1 strategy.

Table 3 presents all results for the first experiment using 25000 calls to the objective function. In the Rosenbrock function, HS reaches the best results, followed by the hybrid algorithm. In the Sphere function, the best results were obtained by DE1, followed by the HS-DE algorithm. The hybrid algorithm got the best results in the Schwefel function, followed by the canonical HS. In the Rastrigin function, the best result was obtained by the HS, followed by HS-DE. Finally, in the Griewank, the minimum was reached by the HS-DE and DE1.

Table 4 shows the results for the experiments using 50000 calls to the objective function. In the Rosenbrock function, the best results were presented by the HS algorithm, followed by the hybrid one. In the Sphere function, the best outcomes were introduced by the Hybrid algorithm and DE1. The DE1 and HS-DE reached the best outcomes in the Schwefel function. In the Rastrigin, the best result came from the HS algorithm. Finally, in the Griewank, the best results were obtained by DE1 and HS-DE.

As previously mentioned, all algorithms have been executed for 50 trials. Thus, according to the central limit theorem, the distribution of data is normal; therefore, it is possible to use parametric tests such as Analysis of Variance (ANOVA). In this context, Table 5 presents all Fs and the \(F_{\text{critical}}\), since if \(F\) is within the interval \(-F_{\text{critical}} < F > F_{\text{critical}}\), the differences between algorithms there exist. Therefore, as presented in Table 5, the differences are meaningful.

Because a complete Tukey test on all functions would demand twelve tables, Table 6 illustrates only those comparisons in which there are no meaningful differences. As we can see from the table, all in all, DE1, HS, and HS-DE presents similar results.

Figure 7 shows the mean of the time for each algorithm on each function in seconds. As we can see, the canonical algorithms execute faster than the harmony search and the hybrid one. However, the hybrid one takes advantage of the faster canonical algorithm, and it is faster than the canonical harmony search.

The stability of the algorithms for 25000 calls to objective function is shown in Figures 8a to 8e. As we can see, the algorithm HS-DE presents higher stability because it presents similar outcomes in all trials. The Harmony Search algorithm also presents a similar behavior. Therefore, we believe that the HS algorithm
<table>
<thead>
<tr>
<th>GA</th>
<th>GA-E</th>
<th>PSO</th>
<th>DE1</th>
<th>DE2</th>
<th>HS</th>
<th>HS-DE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Best</td>
<td>582.1875</td>
<td>61.5629</td>
<td>113.6736</td>
<td>24.8518</td>
<td>5.9320</td>
<td>15.7955</td>
</tr>
<tr>
<td>Mean</td>
<td>126437.6322</td>
<td>182.5494124</td>
<td>491488.074</td>
<td>572.9022</td>
<td>693677.3678</td>
<td>75.3831</td>
</tr>
<tr>
<td>Std-Dev</td>
<td>497630.5205</td>
<td>58.5382</td>
<td>1930499.002</td>
<td>3791.6089</td>
<td>2351928.67</td>
<td>35.251</td>
</tr>
<tr>
<td>Rosenbrock</td>
<td>3.4037</td>
<td>0.1789</td>
<td>0.4078</td>
<td>1.80E-14</td>
<td>5.6906</td>
<td>0.0039</td>
</tr>
<tr>
<td>Mean</td>
<td>6.1353</td>
<td>0.4005</td>
<td>3.3937</td>
<td>2.36E-05</td>
<td>15.7020</td>
<td>0.0064</td>
</tr>
<tr>
<td>Std-Dev</td>
<td>1.7784</td>
<td>0.1205</td>
<td>2.4614</td>
<td>2.02E-04</td>
<td>5.1165</td>
<td>0.012</td>
</tr>
<tr>
<td>Sphere</td>
<td>429.1366</td>
<td>65.0257</td>
<td>22.4713</td>
<td>1.55E+03</td>
<td>3324.1440</td>
<td>13.9350</td>
</tr>
<tr>
<td>Mean</td>
<td>928.2821</td>
<td>170.5744</td>
<td>4426.2282</td>
<td>572.9022</td>
<td>4530.9539</td>
<td>27.2608</td>
</tr>
<tr>
<td>Std-Dev</td>
<td>242.0254</td>
<td>41.7506</td>
<td>2297.9037</td>
<td>6.14E+02</td>
<td>598.9474</td>
<td>9.0918</td>
</tr>
<tr>
<td>Schwefel</td>
<td>28.2776</td>
<td>9.1238</td>
<td>90.6538</td>
<td>55.4900</td>
<td>13.9350</td>
<td>9.45</td>
</tr>
<tr>
<td>Mean</td>
<td>48.7837</td>
<td>15.7264</td>
<td>223.5985</td>
<td>2.86E-05</td>
<td>15.7020</td>
<td>0.0064</td>
</tr>
<tr>
<td>Std-Dev</td>
<td>9.2960</td>
<td>2.9448</td>
<td>57.5209</td>
<td>9.45</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rastrigin</td>
<td>11.6580</td>
<td>1.5884</td>
<td>90.6538</td>
<td>55.4900</td>
<td>13.9350</td>
<td>9.45</td>
</tr>
<tr>
<td>Mean</td>
<td>21.3854</td>
<td>2.3396</td>
<td>1.3454</td>
<td>0.0000</td>
<td>14.7340</td>
<td>0.0032</td>
</tr>
<tr>
<td>Std-Dev</td>
<td>5.8630</td>
<td>0.3864</td>
<td>0.3662</td>
<td>0.0179</td>
<td>19.7671</td>
<td>0.0836</td>
</tr>
<tr>
<td>Griewank</td>
<td>11.6580</td>
<td>1.5884</td>
<td>90.6538</td>
<td>55.4900</td>
<td>13.9350</td>
<td>9.45</td>
</tr>
<tr>
<td>Mean</td>
<td>21.3854</td>
<td>2.3396</td>
<td>1.3454</td>
<td>0.0000</td>
<td>14.7340</td>
<td>0.0032</td>
</tr>
<tr>
<td>Std-Dev</td>
<td>5.8630</td>
<td>0.3864</td>
<td>0.3662</td>
<td>0.0179</td>
<td>19.7671</td>
<td>0.0836</td>
</tr>
</tbody>
</table>

enhances the stability of the hybrid algorithm. The experiments using 50000 calls presented similar results.

5 CONCLUSIONS
This paper presented a study involving a cooperative hybrid algorithm based on Harmony Search and Differential Evolution. Results indicated that the HS-DE algorithm presents, all in all, similar outcomes such as Differential Evolution and Harmony Search, excepting in Schwefel function using 25000 calls to the objective function in which the hybrid algorithm presents the best result. Furthermore, the HS-DE algorithm showed satisfying stability concerning the execution if compared with the other algorithms.
Table 5: ANOVA test considering all algorithms

<table>
<thead>
<tr>
<th>25000 calls</th>
<th>Sphere</th>
<th>Schwefel</th>
<th>Rastrigin</th>
<th>Griewank</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>2.997</td>
<td>326.685</td>
<td>243.754</td>
<td>498.560</td>
</tr>
<tr>
<td></td>
<td>50000 calls</td>
<td>119,651</td>
<td>326,075</td>
<td>508,914</td>
</tr>
</tbody>
</table>

Table 6: Tukey test for all algorithms

<table>
<thead>
<tr>
<th>25000 calls</th>
<th>Sphere</th>
<th>Schwefel</th>
<th>Rastrigin</th>
<th>Griewank</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rosenbrock</td>
<td>HS vs DE1, HS vs GA-E, and HS vs HS-DE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sphere</td>
<td>HS vs DE1, HS vs GA-E, and HS vs HS-DE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Schwefel</td>
<td>FS vs DE2, HS vs GA-E, and HS vs HS-DE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rastrigin</td>
<td>GA-E vs DE1, and HS vs HS-DE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Griewank</td>
<td>GA-E vs DE1, HS vs DE1, PSO vs DE1, HS vs GA-E, PSO vs GA-E, HS vs HS-DE, and PSO vs HS</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>50000 calls</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rosenbrock</td>
</tr>
<tr>
<td>Sphere</td>
</tr>
<tr>
<td>Schwefel</td>
</tr>
<tr>
<td>Rastrigin</td>
</tr>
<tr>
<td>Griewank</td>
</tr>
</tbody>
</table>

Figure 7: Time for each algorithm with 50000 function calls in seconds

Future work includes: (i) a study on parameter optimization; (ii) transforming the hybrid algorithm into a parallel one using multi-core and many-core architectures; (iii) improving the algorithm adding adaptive or self-adaptive features; and (iv) using the algorithm to solve real-world problems such as portfolio investment optimization and dynamic economic dispatch of power.

REFERENCES

Figure 8: Best results in 50 trials per function - 25000 calls