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 Luis, Maranhão, Brasil
sougusta@gmail.com

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

ABSTRACT

Hybrid algorithms aim to mix features from two or more evolutionary/swarm algoprove both the exploration and exploitation abilities of the algorithm. Generally, hybrid algorithms prrithms to imesent 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 Searh, 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 $x \in \mathbf{R}^n$ 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 then 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).

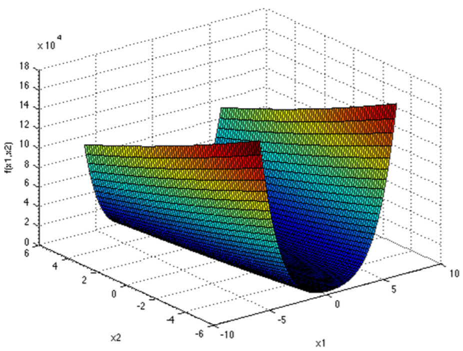


Figure 1: Rosenbrock

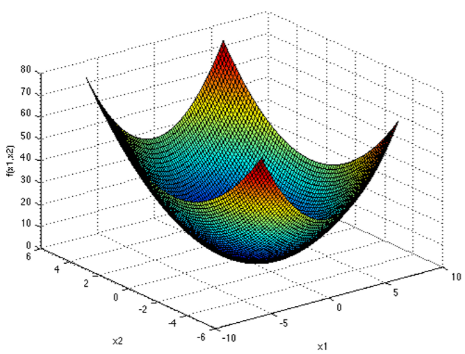


Figure 2: Sphere

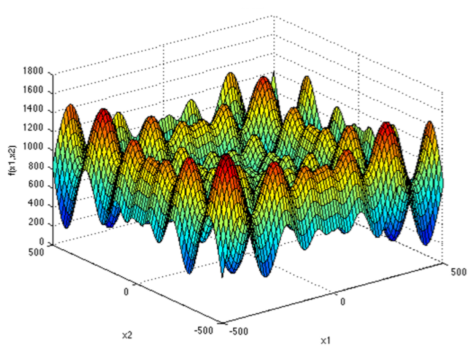


Figure 3: Schwefel

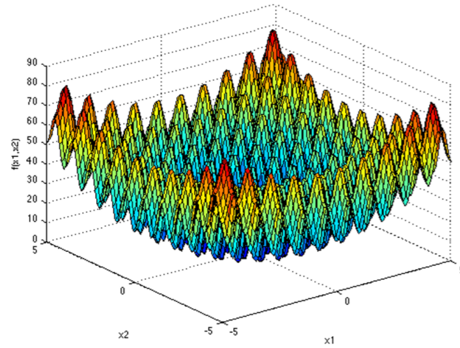


Figure 4: Rastrigin

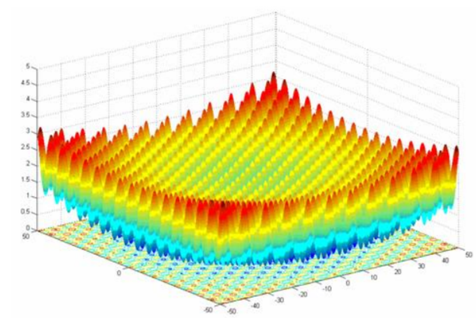


Figure 5: Griewank

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

Table 1: Benchmark functions properties

Name	Function	Domain	Min	Separable	Multimodal	Differentiable
Rosenbrock	$f_1(x) = \sum_{i=1}^{n} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	[-2.048,2.048]	0	No	No	Yes
Sphere	$f_2(x) = \sum_{i=1}^{n} x_i^2$	[-5.12,5.12]	0	Yes	No	Yes
Schwefel	$f_3(x) = \sum_{i=1}^{n} -x_i \sin\sqrt{\lvert x_i \rvert}$	[-500,500]	0	Yes	Yes	Yes
Rastrigin	$f_4(x) = 10n + \sum_{i=1}^{n}(x_i^2 - 10\cos(2\pi x_i))$	[-5.12,5.12]	0	Yes	Yes	Yes
Griewank	$f_5(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}})$	[-600,600]	0	No	Yes	Yes

Algorithm 1 - HS Pseudo Code

HM ⟵ InitHarmonyMemory();
fitness ⟵ Eval(pop);
while stop Criterion not reached **do**
  Improvise a new harmony
  Update the harmony memory
end while

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.

$$x_i' = \begin{cases} x_i \in x_i^1, x_i^1, ..., x_i^{HMS} & , HMCR \\ x_i \in X_i & , 1 - HMCR \end{cases} \tag{1}$$

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.

$$x_i' = \begin{cases} \text{Yes} & , PAR \\ \text{No} & , 1 - PAR \end{cases} \tag{2}$$

If the decision is Yes, then the pitch is adjusted as presented in the Algorithm 2, in which $bandwidth = 0.05$. Finally, if the new harmony is better than the worst one in the HM, then the new one replaces it.

Algorithm 2 - HS Pitch Adjustment

r = random(0,1)
if (r < PAR) **then**
  r = random(0,1)
  if (r < 0.5) **then**
    $x_i' \longleftarrow x_i' - r * bandwidth$
  else
    $x_i' \longleftarrow x_i' + r * bandwidth$
  end if
end if

3.2 Differential Evolution

Differential Evolution (DE) is a metaheuristic developed by Storn e Price [5] in 1995. It works similarly to a Genetic Algorithm, however, using distinct operators. The Algorithm 3 shows its pseudo code.

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 pop_{ij}. Finally, if the new individual is better than that one in the current population, the new one replaces it.

Algorithm 3 - DE Pseudo Code

pop ⟵ InitPopulation();
fitness ⟵ Eval(pop);
while stop Criterion not reached **do**
  Select 3 individuals randomly: $indiv_1$, $indiv_2$, $indiv_3$;
  $v_j \longleftarrow indiv_3 + F \times (indiv_1 - indiv_2)$;
  if (rand() < CR) **then**
    $new\_indiv_j \longleftarrow v_j$
  else
    $new\_indiv_j \longleftarrow pop_{ij}$
  end if
  if fitness(new_indiv) better than fitness(pop_i) **then**
    $pop_i \longleftarrow new\_indiv$;
  end if
end while

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 $indiv_1$ 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

HM ← InitHarmonyMemory(HMS)
while stop Criterion not reached **do**
  $HM' \leftarrow HarmonySearch(HM)$
  $DE' \leftarrow DifferentialEvolution(HM)$
  $Pop \leftarrow Join(HM, HM', DE')$
  $fit \leftarrow Evaluate(Pop)$
  $HM \leftarrow Best(fit, HSM)$
end while

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.

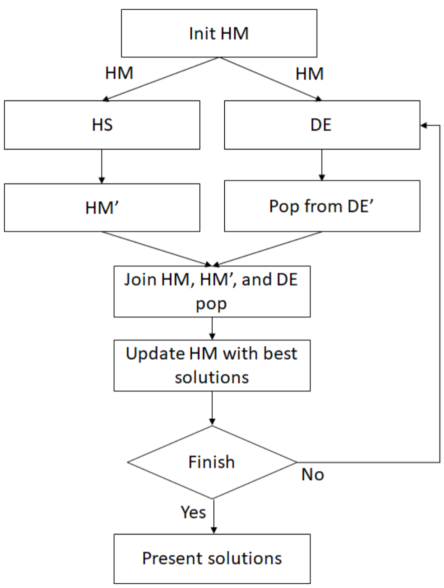


Figure 6: HS-DE Flowchart

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 testes with and without elitism. Regarding DE, we used two strategies: DE/Rand/1 e 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.

Table 2: Parameters used in the algorithms

GA	$p_c = 0.8$, $p_m = 0.05$, with and without elitism
PSO	$w = 0.7$, $c_1 = c_2 = 2.0$, $v_{min} = -100$, $v_{max} = 100$
DE	$F = 0.3$, $p_c = 0.6$, DE/Rand/1, and DE/Best/1
HS	$PAR = 0.3$, $HMCR = 0.95$, $bandwith = 0.05$
HS-DE	$PAR = 0.3$, $HMCR = 0.95$, $bandwith = 0.05$, $F = 0.3$, $p_c = 0.6$, DE/Rand/1

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 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 $F_{critical}$, since if F is within the interval $-F_{critical} > F > F_{critical}$, 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 3: Results for 25000 calls

	GA	**GA-E**	**PSO**	**DE1**	**DE2**	**HS**	**HS-DE**
Rosenbrock							
Best	582.1875	61.5629	113.6736	24.8518	684.2337	**5.9320**	15.7955
Mean	216437.6322	182.5494124	491488.074	572.9022	693677.3678	75.3831	611.9482
Std-Dev	497630.5205	58.5382	1930499.002	3791.6089	2351928.67	35.251	3853.1485
Sphere							
Best	3.4037	0.1789	0.4078	**1.80E-14**	5.6906	0.0039	4.43E-11
Mean	6.1353	0.4005	3.3937	2.86E-05	15.7020	0.0064	5.57E-05
Std-Dev	1.7784	0.1205	2.4614	2.02E-04	5.1165	0.0012	1.68E-04
Schwefel							
Best	429.1366	65.0257	22.4713	1.55E+03	3324.1440	13.9350	**1.13E-01**
Mean	928.2821	170.5744	4426.2282	2.94E+03	4530.9539	27.2608	9.85
Std-Dev	242.0254	41.7506	2297.9037	6.14E+02	598.9474	9.0918	9.45
Rastrigin							
Best	28.2776	9.1238	90.6558	40.5120	55.4490	**2.6637**	3.9885
Mean	48.7837	15.7264	223.5985	65.5777	102.6393	5.8837	9.2040
Std-Dev	9.2960	2.9448	57.5209	9.5012	26.2267	1.9777	2.5646
Griewank							
best	11.6580	1.5884	1.0733	**0.0000**	21.3658	1.0535	**0.0000**
mean	21.3854	2.3396	1.4330	0.0043	60.3952	1.1264	0.0385
std	5.8630	0.3864	0.3662	0.0179	19.7671	0.0437	0.0836

Table 4: Results for 50000 calls

	GA	GA-E	PSO	DE1	DE2	HS	HS-DE
Rosenbrock							
Best	451.0058	39.2119	59.1954	18.4269	1215.2040	**1.7942**	8.5430
Mean	1283.8968	125.6427	981.6143	35.4995	4378.2509	45.5728	55.2769
Std-Dev	458.8725	34.5220	1245.3798	15.9936	2354.3686	37.3818	29.3696
Sphere							
best	2.0995	0.0698	0.0642	**0.0000**	6.2059	0.0019	**0.0000**
mean	6.1717	0.1221	1.3454	0.0000	14.7340	0.0032	0.0001
Std-Dev	1.6591	0.0435	1.5613	0.0000	5.2904	0.0005	0.0001
Schwefel							
best	469.4378	26.8759	28.1167	**0.0004**	3080.7350	0.7181	**0.0004**
mean	921.0091	54.5521	4009.0481	366.9741	4417.5861	3.6995	1.0873
std	238.0650	16.4075	1505.9910	183.8218	518.7798	2.0651	1.6588
Rastrigin							
best	28.8710	3.8125	20.2391	17.5648	50.2783	**0.6137**	3.0812
mean	49.2233	7.0679	159.0209	30.1533	99.9818	2.4452	6.6461
std	9.0091	1.7108	76.5492	6.3162	25.4143	1.3734	1.8477
Griewank							
best	8.2081	1.2397	0.7227	**0.0000**	22.3064	0.5119	**0.0000**
mean	22.1890	1.4192	1.1233	0.0007	51.5851	0.9075	0.0119
std	5.6962	0.1492	0.1571	0.0023	18.1633	0.1327	0.0281

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

$F_{critial} = 2.125$					
25000 calls					
	Rosenbrock	Sphere	Schwefel	Rastrigin	Griewank
F	2.997	332.685	243.754	498.560	416.764
50000 calls					
F	119,651	326,075	508,914	182,514	371,047

Table 6: Tukey test for all algorithms

25000 calls	
Rosenbrock	HS vs DE1, HS vs GA-E, and HS vs HS-DE
Sphere	HS vs DE1, HS vs GA-E, and HS vs HS-DE
Schwefel	PSO vs DE2, HS vs GA-E, and HS vs HS-DE
Rastrigin	GA vs DE1, and HS vs HS-DE
Griewank	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
50000 calls	
Rosenbrock	GA-E vs DE1, HS vs DE1, HS vs GA-E, and HS vs HS-DE
Sphere	GA-E vs DE, HS vs DE1, HS vs GA-E, and HS vs HS-DE
Schwefel	PSO vs DE2, HS vs GA-E, and HS vs HS-DE
Rastrigin	HS vs GA-E, and HS vs HS-DE
Griewank	GA-E vs DE1, HS vs DE1, PSO vs DE1, HS vs GA-E, PSO-GA-E,

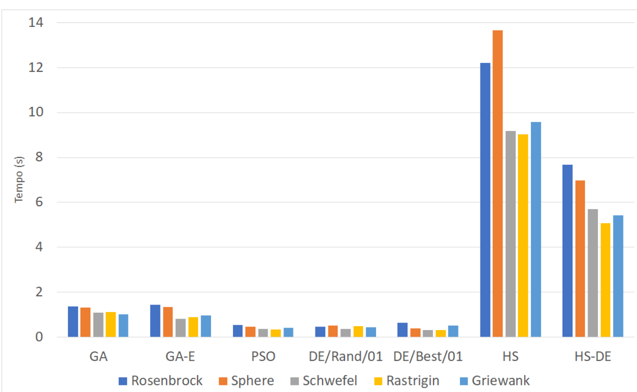
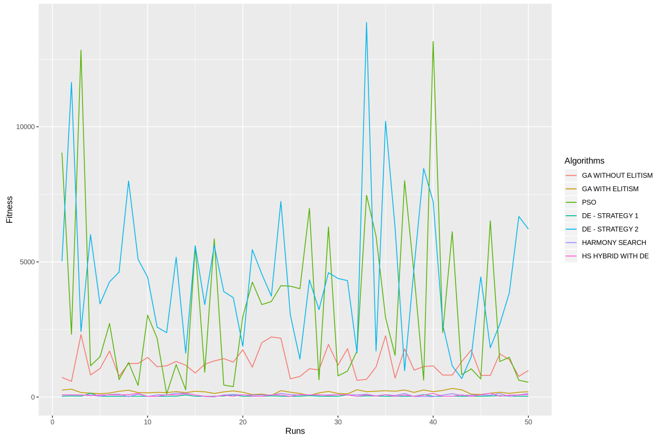


Figure 7: Time for each algorithm with 50000 function calls in seconds

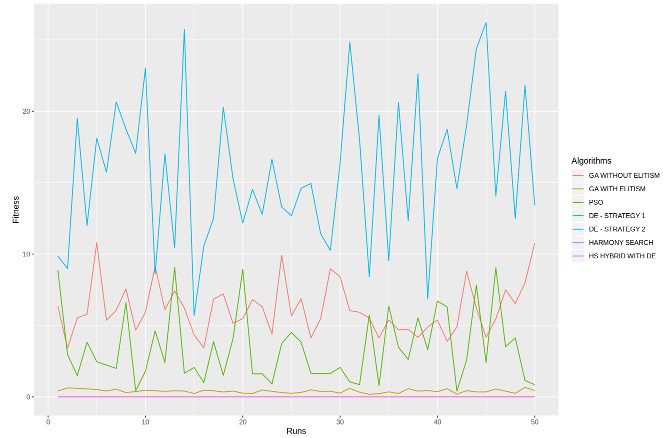
Future work includes: (i) a study on parameter optimization; (ii) transforming the hybrid algorithm into a parallel one using multi-core and many-core architectures; (iii) improving the algorithm adding adaptive or self-adaptive features; and (iv) using the algorithm to solve real-world problems such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

REFERENCES

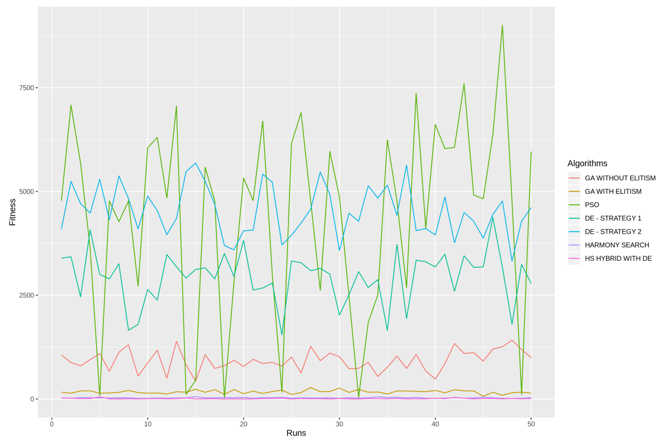
[1] W. Zang, L. Ren, W. Zhang, and X. Liu. A cloud model based dna genetic algorithm for numerical optimization problems. *Future Generation Computer Systems*, 81: 465–477, 2018. ISSN 0167-739X. doi: https://doi.org/10.1016/j.future.2017.07.036.
[2] A. Muc and I. Sanetra. The effectiveness of optimization algorithms in shape and topology discrete optimisation of 2-d composite structures. In *2017 7th International Conference on Modeling, Simulation, and Applied Optimization (ICMSAO)*, pages 1–5, April 2017.
[3] M. Jamil and X-S. Yang. A literature survey of benchmark functions for global optimisation problems. *International Journal of Mathematical Modelling and Numerical Optimisation*, 4(2):1–47, 2013.
[4] H. H. Rosenbrock. An automatic method for finding the greatest or least value of a function. *Computer Journal*, 3(3):175–184, 1960.
[5] R. Storn and K. Price. Differential evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces, 1995.
[6] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proceedings of ICNN'95 - International Conference on Neural Networks*, volume 4, pages 1942–1948, 1995.
[7] Z. Michalewicz, editor. *Genetic Algorithms + Data Structures = Evolution Programs.* Springer, New York, 3 edition, 1999.
[8] Zong Woo Geem, Joong Hoon Kim, and G.V. Loganathan. A new heuristic optimization algorithm: Harmony search. *SIMULATION*, 76(2):60–68, 2001.
[9] R. Thangraj, M. Pant, A. Abraham, K. Deep, and V. Snasel. Differential evolution using a localized cauchy mutation operator. In *2010 IEEE International Conference on Systems, Man and Cybernetics*, pages 3710–3716, Oct 2010.
[10] D. Karaboğa and E. Kaya. Evaluation of performance of adaptive and hybrid abc (aabc) algorithm in solution of numerical optimization problems. In *2018 Innovations in Intelligent Systems and Applications Conference (ASYU)*, pages 1–5, Oct 2018.
[11] W. Zhongyu, L. Yaru, and T. Yingqi. An efficient hybrid de-woa algorithm for numerical function optimization. In *2019 IEEE 28th International Symposium on Industrial Electronics (ISIE)*, pages 2629–2634, June 2019.
[12] K. D. De Jong. An analysis of the behavior of a class of genetic adaptive systems.
[13] H.-P. Schwefel, editor. *Numerical optimization of computer models.* Wiley, 1981.
[14] A. Törn and A. Zilinskas. Lecture notes in computer assisted diagnosis, n. 350, 1989.
[15] A. O. Griewank. Generalized descent for global optimization. *Journal of Optimization*, 34(1):11–39, 1981.
[16] J. Nocedal and S Wright, editors. *Numerical Optimization.* Springer, New York - USA, 2006.
[17] O. A. C. Cortes, A. Rau-Chaplin, and R. F. Lopes. A pso-based algorithm with local search for multimodal optimization without constraints. In *XXXVIII Conferencia Latinoamericana En Informatica (CLEI)*, pages 1–7, Oct 2012.
[18] D. O. Boyer, C. H. Martfnez, and N. G. Pedrajas. Cixl2: A crossover operator for evolutionary algorithms based on population features. *Journal of Artificial Intelligence Research*, 24:1–48, 2005.
[19] Stefan Voß. Meta-heuristics: The state of the art. In Alexander Nareyek, editor, *Local Search for Planning and Scheduling*, pages 1–23, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.
[20] Xin-She Yang. *Harmony Search as a Metaheuristic Algorithm*, pages 1–14. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.



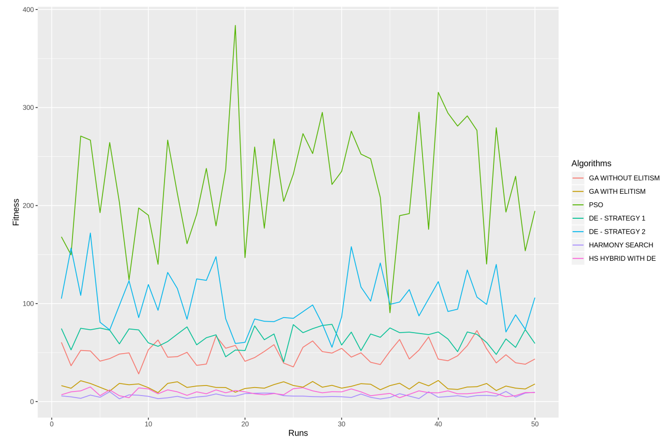
(a) Rosenbrock



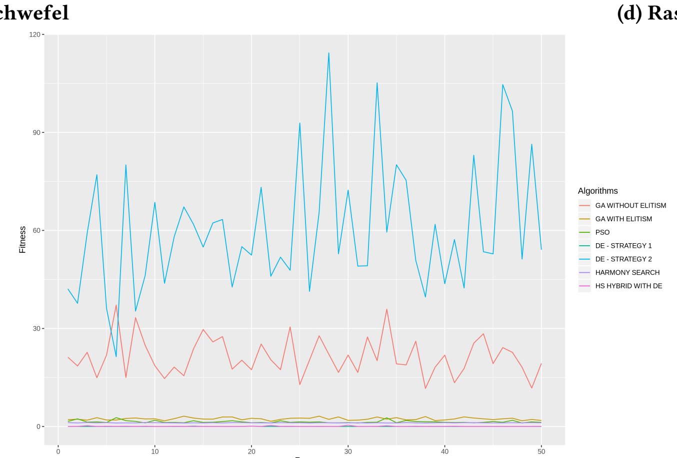
(b) Sphere



(c) Schwefel



(d) Rastrigin



(e) Griewank

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