A PSO-Based Algorithm With Local Search for Multimodal Optimization Without Constraints

Omar Andres Carmona Cortes[1]
Andrew Rau-Chaplin[2]
Rafael Fernandes Lopes[1]
[1]Departamento Acadêmico de Informática
Instituto Federal de Educação, Ciência e Tecnologia do Maranhão
Av. Getulio Vargas, 04 - Monte Castelo
São Luis - MA - 65025-001 - Brazil
Email: {omar, rafaelf}@ifma.edu.br

[2] Faculty of Computer Science, Dalhousie University
6050 University Avenue
Halifax, CA
Email: arc@cs.dal.ca

***Abstract*—The purpose of this paper is to present a PSO algorithm mixed with a new hybrid local search algorithm named LHS, enhancing the exploration and exploitation capabilities of the canonical PSO. The hybrid PSO, named PSOLHS, is examined against six known multimodal functions and compared with both canonical PSO and LHS. Furthermore, a comparison between evolutionary strategies (ES) and MPSO-LS is going to show how our approach outperforms these other techniques in almost all benchmark functions. All comparisons are based on a statistical t-test for supporting our results.**

***Index Terms*—PSO, Hybrid Algorithm, Multimodal,Optimization.**

I. Introduction

The term global optimization refers to the process of attempting to find out the solution x^* of a set of possible solutions S, which has the optimal value for some fitness function f. In other words, we are trying to find out the solution x^* such that $x \neq x^* \Rightarrow f(x^*) \geq f(x)$ [2], in maximization problems, or $x \neq x^* \Rightarrow f(x^*) \leq f(x)$, in minimization problems. In both cases, the best solution is called global optima. Solutions other than global optima are called local optima.

A function to be optimized is monomodal if it has only one global optima and no locals optima. On the other hand, a multimodal function has many locals optima and, normally, one global optima, however, there are multimodal functions with many global optima as well.

Global optimization algorithms, based on mechanisms inherited from biology can be applied effectively when solving continuous problems, specially in difficult cases in which other classical computing strategies, such as gradient, have failed.

Continuous global optimization problems with multimodal objective functions, in which the basins of attraction of local extremes are separated by large areas on which moderate or low objective variability is observed (plateaus), belong to the group of important difficult ones. Another type of difficulty is caused by the low regularity of the objective function when the gradient and the Hessian computation require costly approximative routines or is generally meaningless. [1]

In order to solve multimodal functions, many bio-inspired approaches have been proposed. Particularly, different PSO (Particle Swarm Optimization) algorithms have been used for solving multimodal problems. These algorithms are mostly based on existing approaches used in the evolutionary algorithms [3].

Li et al. [4] investigates some mutations operators based on global best particle. Specifically, three mutations operators are used for helping PSO jump out of local optima: Cauchy, Gaussian and Levy. All mutation operators have an equal initial selection ratio with 1/3 and the mutation operators that result in higher fitness values has its ratio increased.

Pat and Hota [6] propose a modified and improved quantum-behaved particle swarm optimization (QPSO) [11] adding fitness weighted recombination, i.e., recombining two particles using arithmetic crossover, which were chosen by the roulette wheel selection.

Chen [7] uses hill climbing to improve the solution of a particle chosen at random. Then, he modified his basic PSO-LS by choosing the best particles (MPSO-LS) as initial solutions for local search, showing that this last approach presents better results.

Ozcan [3] adds a random walk component and a hill climber to enhance the exploration and exploitation capabilities of PSO. Khairy [5] proposes a PSO algorithm with local search algorithm where a small independent swarms is put around a selected particles within the specified range. In this context, two cases are considered. In the first one, a random particle is chosen for the local search, then the new particle replaces the old one if its fitness is better. In the second one, the new particle replaces the old one based in some probability, just like simulated annealing.

In this work, we combine a new local search algorithm, proposed in the Cortes' work [8], with canonical PSO. This new local search algorithms is also a hybrid algorithm combining features of clonal selection [9], genetic mutation and hill climbing [10], enhancing the capacity of exploration and exploitation of the canonical PSO.

For this sake, the paper is organized as follows: Section II presents the canonical PSO, the used local search algorithms and the proposed one (PSOLHS); Section III shows the benchmarks functions used in our experiments; Section IV discusses the computational experiments; finally, Section V presents the conclusions and future works.

II. The algorithms

A. Particle Swarm Optimization

The algorithm consists of particles that are placed into the search space. Each particle moves combining some aspects of its own history position and the global position. All particles move to next position and probably the swarm moves towards the potential optimum, in the next iteration.

A particle represents a position in the search space as $X_i^D = (x_i^1, x_i^2, ..., x_i^D)$. Further, a particle has a velocity $V_i^D = (v_i^1, v_i^2, ..., v_i^D)$ which is used to determine the new position of the particle in the next iteration, where D represents the problem dimension. The new position is determined by means of the Equations 1 and 2, where W represents the inertia weight, c_1 and c_2 are acceleration constants, r_1 and r_2 are random number in the range $[0, 1]$, p_i^d is the best position of the particle, and g^d is the global optima of the swarm so far.

$$v_i^d = W \times v_i^d + c_1 \times r_1 \times (p_i^d - x_i^j) + c_2 \times r_2 \times (g^d - x_i^d) \quad (1)$$

$$x_i^d = x_i^d + v_i^d \quad (2)$$

The velocity in any dimension is narrowed by a maximum velocity V_{max} in the range $[-V_{max}, V_{max}]$. When the velocity of a particle violates this range we update it using the Equation 3. In other words, we chose a new random velocity within the range $[-V_{max}, V_{max}]$.

$$v_i^d = -V_{max} + (rand() \times (V_{max} - (-V_{max}))) \quad (3)$$

B. Local Hybrid Search - LHS

The Local Hybrid Search was proposed in Cortes' work [8]. Basically, the LHS works according to the pseudocode presented in Figure 1, where s represents the first random solution, P is a population created by cloning c times the s individual, then the population undergo a non-uniform mutation from genetic algorithms and only the best solution goes to the next generation.

```
$s \Leftarrow initialize\_solution(n)$;
$Eval(S)$;
while $NOT(Termination\ Criteria)$ do
    $P \leftarrow Clone(s, c)$;
    $P' \leftarrow Mutation(P)$;
    $s \leftarrow Eval(P')$;
end while
```

Fig. 1. Pseudocode for LHS Algorithm

In the original work, the LHS has some variations, so we used a variation named HNURT (Hybrid Non-Uniform Mutation with Random T) that presented the best results. In this approach, the maximum number of elements to be muted are obtained by the Equation 4, where n is the number of genes/dimension, G_{max} is the max generation number and t is the current generation.

$$nm = rand(\lfloor \frac{n \times (G_{max} - t)}{G_{max}} + l \rfloor) \quad (4)$$

Actually, the Equation 4 is a linear equation that produces a number between 1 and n (only if $G_{max} \geq n$) based on the current iteration. In other words, the number of elements undergo mutation (nm) is a random number belonging to $[1, n]$ in the initial generations and equals 1 in the last one.

Further, a position is mutated using non-uniform mutation according to Equation 5 , where $\tau \in \{0, 1\}$ is randomly generated.

$$c_i^{'} = \begin{cases} c_i + \Delta(t, b_i - c_i), & \text{se } \tau = 0 \\ c_i - \Delta(t, c_i - a_i), & \text{se } \tau = 1 \end{cases} \quad (5)$$

The Δ function is calculated by Equation 6, where r is a random number belonging to $[0, 1]$, b is a parameter chosen by user, t is the current generation and G_{max} is the maximal generation number. This equation is similar to non-uniform mutation of genetic algorithms excepting by the constant 0.5.

$$\Delta(t, y) = y(1 - r^{(1-0.5\frac{t}{G_{max}})^b}) \quad (6)$$

In this context, we can notice that non-uniform mutation combined with nm tries to explore the search space in the early iterations whereas exploit it in the final ones. All in all, the probability of Δ function returns a number close to zero increases as the algorithm advance.

C. The Proposed Algorithm - PSOLHS

The proposed algorithm uses the LHS for enhancing the exploration and exploitation capabilities of PSO, applying the LHS in the best particle according to the pseudocode presented in Figure 2.

```
$X \leftarrow initialize\_swarm(n)$;
$V \leftarrow initialize\_velocities(n)$;
while $NOT(Termination\ Criteria)$ do
  $eval(X)$;
  $s \leftarrow select\_one\_particle(X)$
  $xl \leftarrow HNURT(s, K)$
  if $Fitness(xl) < Fitness(s)$ then
    $p \leftarrow xl$
  end if
  if $Fitness(xl) < gbest$ then
    $gbest \leftarrow xl$
  end if
  $V \leftarrow update\_velocity(V)$;
  $X \leftarrow update\_positions(X)$;
end while
$eval(X)$;
if $min(Fitness(X)) < gbest$ then
  $gbest \leftarrow x$
end if
```

Fig. 2. Pseudocode for PSOLHS

The proposed algorithm tries to enhance the fitness of the best particle using LHS algorithm before the swarm search, where K is the number of iterations that the LHS can use for. We have been applied the local search to the best particle because Chen [7] proved to be the best choice. Afterwards, the algorithm executes the canonical PSO, updating the position and the velocity of particles. Indeed, this particular order provides a better initial solution to the canonical PSO.

III. Benchmark Functions

In this work, we have used some known multimodal benchmarks function to test if the performance of an algorithm is better than another. In all presented cases the benchmark functions must be minimized.

Each function has basically two important features: separability and multimodality. A function is multimodal if it has two or more local optima. It is separable if it can be rewritten as a sum of p functions of just one variable. Non separable functions are more difficult to optimize because the accurate search direction depends on two or more elements of the solution vector. The problem is even more difficult if the function is also multimodal.

The dimensionality of the search space is another important factor in the problem complexity, because the number of local optima increases with the problem dimension. A study of the dimensionality problem and its features was carried out by Friedman [12]. We have used 6 benchmarks functions as shown in Table I, where all of them are multimodals. The domain of each function, the global optima, its name and whether the function is separable or not is presented, as well.

The Schwefel's function, also known as Surface Schwefel, is composed of a great number of peaks and valleys. The function has a second best minimum far from the global minimum where many search algorithms are trapped. Furthermore, the global minimum is near the bounds of the domain.

The Rastrigin's function was constructed from Sphere ($f(x) = \sum x^2$) and its contour is made up of a large number of local minima, whose value increases with the distance to the global minimum. Therefore, the algorithm could be trapped if genes are produced far from the global optima but into domain.

The Ackley's function has an exponential term that covers its surface with numerous local minima. The complexity of this function is moderated, thus the search strategy must combine the exploratory and exploitative components efficiently.

The Griewank's has a product term that introduces interdependence among the variables. The aim is the failure of the techniques that optimize each variable independently. As in Ackley's function, the optima of Griewank's function are regularly distributed. Finally, Penalized functions, also known as Generalized Penalized Functions, are similar to Ratrigin's function, nonetheless, penalized functions are no separable.

IV. Experimental Results

The experiments have been conducted in a Intel i5 processor 2.67Ghz, 4GB (RAM), 500GB (HD) and Matlab 2010. In order to compare the algorithms variations, we solved each benchmarks through 31 independent trials and dimension equals 30 ($n = 30$). This number of executions is based on the central limit theorem, which claims that with 30 or more runs a sample will present a normal distribution, allowing statistical inferences [13], like a t-test, for instance.

Considering a t-test, our null hypothesis is that there are no differences between means, i.e., $H_0 : \mu_1 = \mu_2$, which indicates a two tailed test with level of significance $\alpha = 0.5$ with 30 degrees of freedom. Therefore, any result of t, calculated by Equation 7, has to be in the range [-2.042, 2.042] for accepting the null hypothesis, where μ_i are the means being compared, σ_i are the standard deviations and n is the number of trials.

$$t = \frac{\mu_1 - \mu_2}{\sqrt{\frac{\sigma_1^2}{n} + \frac{\sigma_2^2}{n}}} \tag{7}$$

Moreover, in order to make a fair comparison between algorithms, the same number of functions evaluations have been done. For instance, the f_1 function executes 9000 iterations in PSO and LHS in each trial, then the PSO-LHS executes 1800 iterations times 5 for the local search ($1800 \times 5 = 9000$). All experiments have been conducted using the parameters presented in Table II, where ub and lb are the upper bound and the lower bound of the search space, respectively.

The Figure 3 presents the performance of PSO, LHS and PSOLHS algorithms. Clearly, PSO has got trapped in

TABLE I
BENCHMARKS FUNCTIONS

Benchmark Functions	Domain	Min	Name	Separable
$f_1(x) = \sum_{i=1}^{n} -x_i \times \sin(\sqrt{x_i})$	$[-500, 500]$	-12569.5	Schwefel's	yes
$f_2(x) = \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i) + 10]$	$[-5.12, 5.12]$	0	Rastrign's	yes
$f_3(x) = -20\exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}\cos x_i^2} - \exp(\frac{1}{n}\sum_{i=1}^{n}\cos 2\pi x_i)$ $+20 + e$	$[-32, 32]$	0	Ackley's	no
$f_4(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}})$	$[-600, 600]$	0	Griewank's	no
$f_5(x) = \frac{\pi}{n}\{\sin^2(\pi y_1) + \sum_{i=1}^{n-1}(y_i - 1)^2[1 + 10\sin^2(\pi y_{i+1})]$ $+(y_n - 1)^2\} + \sum_{i=1}^{n} u(x_i, 10, 100, 4)$ $y_i = 1 + \frac{1}{4}(x_i + 1)$ $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, & x_i > a \\ 0, & -a < x_i < a \\ k(-x_i - a)^m, & x_i < -a \end{cases}$	$[-50, 50]$	0	Penalized	no
$f_6(x) = 0.1\{\sin^2(3\pi x_1) + \sum_{i=1}^{n-1}(x_i - 1)^2[1 + \sin^2(3\pi x_{i+1})] +$ $(x_n - 1)[1 + \sin^2(2\pi x_n)]\} + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$	$[-50, 50]$	0	Penalized	no

TABLE II
PARAMETERS FOR PSO, PSOLSH

PSO/PSOH		LHS/GA	
#particles	100	#clones	100
c_1, c_2	1.49618	#generations	5
W	0.72984	b	5
V_{max}	$\frac{ub-lb}{2}$		

a local mínima in f_1, f_2 and f_3, and seems to have similar performance in f_4, f_5 and, specially, in $f6$. However, looking into the numbers in Tables IV and Table V we can notice that PSOLHS also outperformed the other algorithms in f_5 and $f6$ and, on the other hand, we have to accept the null hypothesis in f_4.

In f_2 LHS seems to evolve faster than PSOLHS, however, PSOLHS ends up with better results after about 700 iteration. This behavior is inverted in f_3 where PSOLHS evolves faster than LHS, probably because f_3 is more complex and no separable. The same idea is repeated in f_6 for the canonical PSO, but looking at Table IV we can observe that PSO is also trapped in a local minima while in PSOLHS the evolving process carried out.

Table III presents the success rate of the local search according to the number of iterations (#Iterat.). The column Rate represents the mean of the number of times that the local search in PSOLHS enhances the solution. The column Last It. presents in which iteration the enhanced of the solution happened for the last time. The last improvement of the solution by the local search is illustrated in the column Last Solution. The last column shows the mean accomplished by PSOLHS.

In this context, we can notice that the final solution was obtained practically in the iteration 461 in the function f_1, becoming the searching for better solutions a hard task. The exploration capability is clear in function f_2. On the other hand, the LHS take the advantage of the exploitation capability in functions f_3, f_4, f_5 and f_6 because the solutions were enhanced close to the final iteration (#It versus Last It.), and the differences between the last solution, improved by local search, and the final one (Mean of PSOLHS) is not far-away.

TABLE III
SUCCESS RATE OF THE LOCAL SEARCH

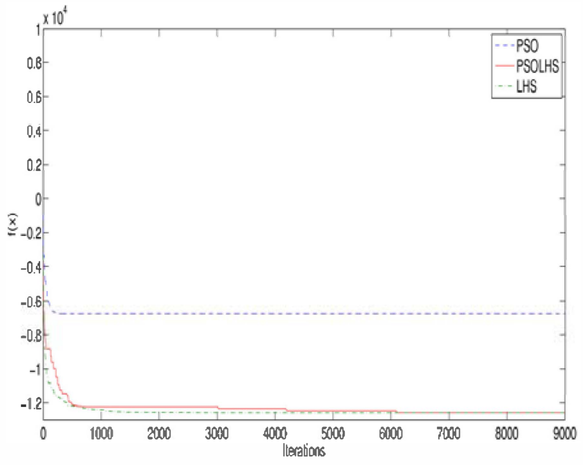
F.	#It.	Rate	Last It.	Last Solution	Mean of PSOLHS
f_1	1800	259.3	461	-12569.48662	-12569.48662
f_2	1000	284.8	300	2.68E-06	2.42043E-13
f_3	300	130.1	222	0.000940578	8.78284E-05
f_4	400	139.9	194	0.05143648	0,01547516
f_5	300	147	240	2.61E-08	5,47554E-09
f_6	300	147.9	261	2.56E-06	3,06636E-07

Table IV shows a closer comparison between the canonical PSO and PSOLHS, where the second column represents the number of iterations of each algorithms which were chosen based on Yao's work [14]. So, taking the t-test into account, the function f_4 has the same behavior in both algorithms (accepting the null hypothesis), probably because the search space is quite huge and the function is not separable, consequently, becoming the function very hard to optimize. The PSOLHS clearly outperformed the results of the canonical PSO, in the other function.

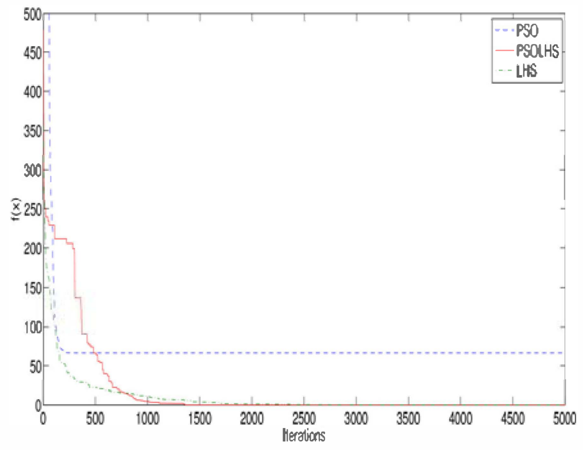
TABLE IV
COMPARISON BETWEEN PSOLHS AND PSO

	#Iter.	PSO	PSOLHS	t
f_1	9000	-6976.8551 (519,545068)	-12569.4866 (5,2147E-07)	59.9340777
f_2	5000	55.4608487 (15.2414901)	2.4204E-13 (3.9627E-13)	20.26002274
f_3	1500	2.39334217 (1.33037127)	8.7828E-05 (6.2498E-05)	10.01605827
f_4	2000	**0.02670347** (0.03460991)	**0.01812523** (0.02005646)	**1.194001145**
f_5	1500	1.07585499 (2.60960767)	5.4755E-09 (1.6195E-08)	2.295405212
f_6	1500	1.88703848 (4.13262833)	3.0664E-07 (7.7303E-07)	2.542349096

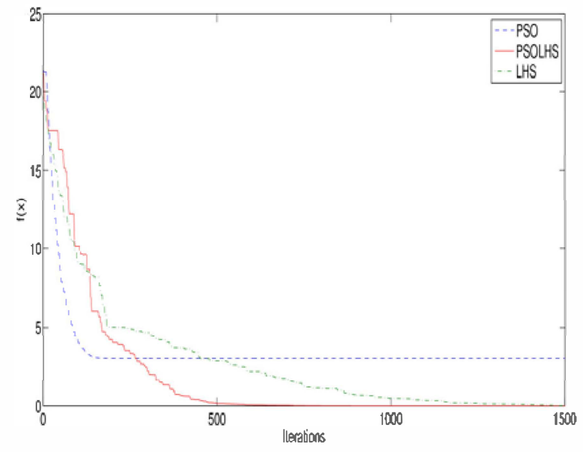
Table V presents a comparison between PSOLHS and pure LHS. Again, the PSOLHS algorithm overcame the



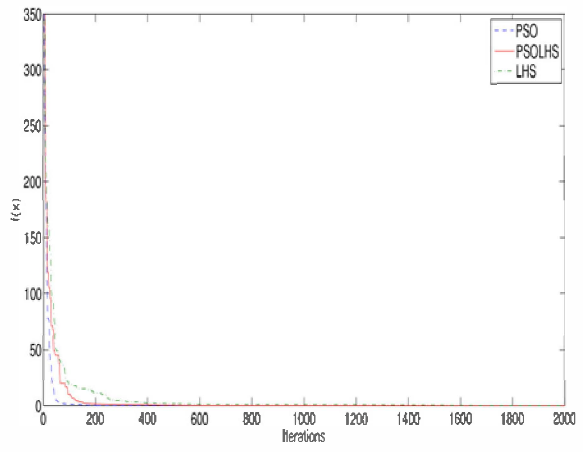
f_1



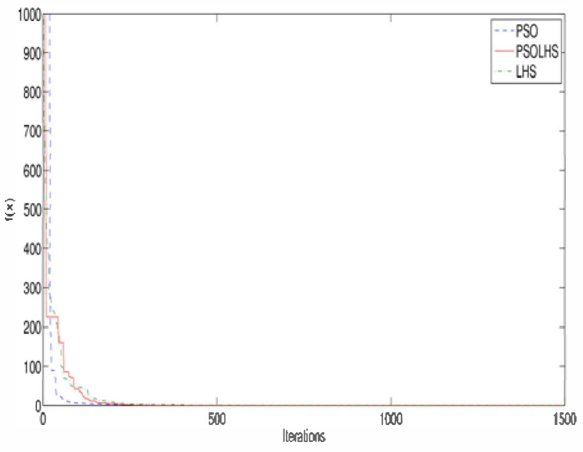
f_2



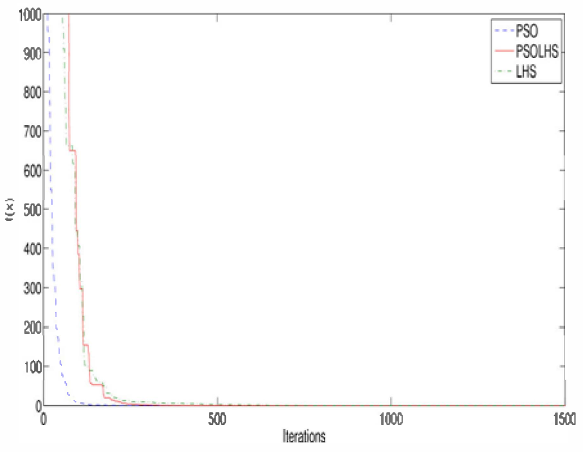
f_3



f_4



f_5



f_6

Fig. 3. Performance of the PSO, LHS and PSOLHS (f_1 to f_6)

results of LHS, excepting in function f_4, probably for the same reason explained earlier.

TABLE V
Comparison between PSOLHS and LHS

	#Iter.	LHS	PSOLHS	t
f_1	9000	-12569.48599 (0.000186)	-12569.48662 (5.21472E-07)	18.71078625
f_2	5000	0.000183 (0.000058)	2.42043E-13 (3.96268E-13)	17.5672565
f_3	1500	0,020361 (0.005517)	8.78284E-05 (6.24979E-05)	20.45840137
f_4	2000	**0.027999** (0.018242)	**0.018125226** (0.020056458)	**2.027733631**
f_5	1500	0.000011 (0.000007)	5.47554E-09 (1.61953E-08)	8.744965381
f_6	1500	0.000639 (0.000773)	3.06636E-07 (7.73029E-07)	4.600378234

Indeed, the results presented by PSOLHS are very good, because the standard deviation is small in all benchmark functions, representing a good behavior of the algorithm, which means that the PSOLHS is able to find out good solutions in all trials.

A. Comparison With Evolution Strategies

Table VI compares the PSOLHS with evolution strategies (ES), which results were obtained from Yao's work [14]. We run the hybrid algorithm 31 times and Yao runs his algorithm 50 times, therefore the *t-test* has 79 degree of freedom and significant $\alpha = 0.5$ by a two-tailed test, i.e., $-1.99 \leq t \leq 1.99$ for accepting the null hypotheses.

TABLE VI
Comparison between PSOLHS and EE

	#Iter.	PSOLHS	ES	t
f_1	9000	**-12569.48662** (5,21472E-07)	**-12569.47995** (52,6)	**0.000905314**
f_2	5000	2.42043E-13 (3.96268E-13)	0.046 (0.12)	2.737547564
f_3	1500	8.78284E-05 (6.24979E-05)	0.018 (0.0021)	60.86923608
f_4	2000	**0,018125226** (0.020056458)	**0,016** (0.022)	**-0,448372075**
f_5	1500	5.47554E-09 (1.61953E-08)	0.000009 (0.000004)	16.05822164
f_6	1500	3.06636E-07 (7.73029E-07)	0.00016 (0.000073)	15.62100755

Considering the *t-test* between PSOLHS and ES, we have to accept our null hypothesis in functions f_1 and f_4, so we can state that these particular functions have the same behavior in both algorithms. However, the PSOLHS presents a better standard deviation in f_1, therefore it has a better stability with this generation number, meaning the PSOLHS find out good solutions in all trials. Again, PSOLHS outperformed the ES in functions f_2, f_3, f_5 and f_6.

B. Comparison With PSO-LS

The algorithm PSO-LS, proposed by Chen [7], adds a local search algorithm (hill climbing) before it communicates information with other particles in the swarm. Specifically, we are using the version named MPSO-LS where the local search is applied only to the best particle, exactly as we did in PSOLHS. The population size is 80 particles and 2000 iterations. We are going to present the mean of 30 trials for all experiments.

The comparison is done using functions f_2 (Rastrign), f_4 (Griewank) and Rosembrock, which we called as f_7 according to Equation 8. Despite this function being a monomodal, it is hard to solve due to its quadratic features. Moreover, there are some discussions claiming that the Rosembrock's function is multimodal as well when its dimension is greater than three [15].

$$f_7 = \sum_{i}^{n} (100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2) \tag{8}$$

The search space of the functions and the initialization range are introduced in Table VII. As we can observe, the initialization range introduces a new difficult to the algorithms, because these ranges produce higher outcomes to the functions in the earlier iterations, becoming the function harder to solve.

TABLE VII
Search Space and Initialization Range

Function	Search Space	Initialization Range
f_7	[-100,100]	[15, 30]
f_2	[-10,10]	[2.56, 5.12]
f_4	[-600,600]	[300, 600]

Unfortunately, we cannot uses a t-test like we did before, because the Chen's work does not present standard deviation. Nonetheless, the comparison is possible because the PSOLHS presents much better results for f_2 and f_7 as we can see in Table VIII. On the other hand, the results are quite similar in f_4, indicating the same behavior of the previous algorithms.

TABLE VIII
Comparison between MPSO-LS and PSOLHS

Function	MPSO-LS	PSOLHS
f_7	**78,7998**	**3,90239038**
f_2	**28,8657**	**4,03405E-13**
f_4	0,012	0,011194731

V. Conclusions

In this paper we proposed an effective new hybrid algorithm combining a canonical PSO with a local search (LHS) algorithm for optimization of global/multimodal functions. Further work needs to be done on testing more multimodal functions. For instance, the improvements of the exploitation capabilities have to be investigated in the LHS algorithm.

We have confirmed from the results of simulation experiments that using benchmark problems the proposed method has superior search capabilities in comparison with canonical PSO, LHS, ES and MPSO-LS.

Further challenges for the future include optimization using automatic adaptation of the local search algorithms, applying a Simulated Annealing exchange policy and the use of fuzzy controllers to adjust parameters in real-time execution.

References

[1] Schaefer, R., "Foundation of Genetic Global Genetic Optimization", Studies in Computational Intelligence, v 74, Berlim: Springer Verlag, 2007.

[2] Eiben, A. E.; Smith, J. E., "Introduction to Evolutionary Computation", 2ed, Berlim: Springer Verlag, 2007.

[3] Ozcan,E.; Yilmaz, M. "Particle Swarms for Multimodal Optimization", *Proc. SProceedings of the 8th international conference on Adaptive and Natural Computing Algorithms, Part I*, pp. 366-375, 2007.

[4] Li, C.; Yang, S.; Korejo, I., "An Adaptive Mutation Operator for particle swarm optimization" *Proc. VIII Metaheuristics International Conference*, pp. 13-16, 2008.

[5] Khairy, M.; Fayek, M.B.; Hemayed, E.E., "PSO2: Particle swarm optimization with PSO-based local search", *Proc. IEEE Congress on Evolutionary Computation (CEC)*, pp. 1826-1832, 2011.

[6] Pat, A.; Hota, A. R., "An Improved Quantum-behaved Particle Swarm Optimization Using Fitnessweighted Preferential Recombination", *Proc. Second World Congress on Nature and Biologically Inspired Computing*, Kitakyushu, Fukuoka, Japan, 2010.

[7] Chen, J.; Qin, Z.; Liu, y.; Lu, J., "Particle Swarm Optimization with Local Search", *Proc. International Conference on Neural Networks and Brain*, Beijing, pp. 481-484, IEEE Press, 2005.

[8] Cortes, O.A.C. ; da Silva, J.C., "A Local Search Algorithm Based on Clonal Selection and Genetic Mutation for Global Optimization",*Proc. In Proceeding of Eleventh Brazilian Symposium of Neural Networks (SBRN)*, IEEE Press, 2010.

[9] de Castro, L. N.; Von Zuben, F. J., "Learning and optimization using the clonal selection principle", IEEE Transactions on Evolutionary Computation, vol. 6, pp. 239-251, 2002.

[10] Michalewicz, Z., *Genetic Algorithms + Data Structures = Evolution Programs*, 2 ed, Berlim: Springer Verlag, 1999.

[11] Sun, J.; Feng, B.; Xu, W. B., "Particle swarm optimization with particles having quantum behavior", *Proc. IEEE Proceedings of Congress on Evolutionary Computation*, pp. 325-331, 2004.

[12] Friedman, J. H., An overview of predictive learning and function approximation. *Proc. V. Cherkassky, J. H. Friedman, and H. Wechsler, editors, From Statistics to Neural Networks, Theory and Pattern Recognition Applications*, volume 136 of NATO ASI Series F, pages 1-61. Springer-Verlag, 1994.

[13] Schefler, W. C., Statistics: Concept and Applications, The Benjamin/Cummings Publishing Company, California, 1988.

[14] Yao, X.; Liu, Y.; Lin, G., *Proc. Evolutionary Programming Made Faster*, IEEE Transaction on Evolutionary Computation, v. 3, n. 2, july, 1999.

[15] Tang, K.; Li, X.; Suganthan, P. N.; Yang, Z.; Weise, T., "Benchmark Functions for the CEC'2010 Special Session and Competition on Large-Scale Global Optimization". *Proc. IEEE World Congress on Computational Intelligence.* China, 2009.