Hybrid PSO Algorithm with Iterated Local Search Operator for Equality Constraints Problems

Felipe Mota*, Vinícius Almeida†, Elizabeth F. Wanner‡ and Gladston Moreira*
*Computing Department, Universidade Federal de Ouro Preto, Ouro Preto, MG, Brazil 35400-000
†Department of Mathematic, Universidade Federal de Ouro Preto, Ouro Preto, MG, Brazil 35400-000
‡EAS, Aston University, Birmingham, UK

***Abstract*—This paper presents a hybrid PSO algorithm (Particle Swarm Optimization) with an ILS (Iterated Local Search) operator for handling equality constraints problems in mono-objective optimization problems. The ILS can be used to locally search around the best solutions in some generations, exploring the attraction basins in small portions of the feasible set. This process can compensate the difficulty of the evolutionary algorithm to generate good solutions in zero-volume regions. The greatest advantage of the operator is the simple implementation. Experiments performed on benchmark problems shows improvement in accuracy, reducing the gap for the tested problems.**

***Keywords*—Hybrid, Evolutionary Algorithm, Equalities, Constraints**

I. Introduction

Although Evolutionary Algorithms (EA's) are a powerful tool in optimization, they have been originally developed to solve unconstrained problems [17]. When it comes to constrained problems, the randomness of the algorithm may lead into unfeasible areas, mainly when they have equality constraints, resulting in zero-volume objects [15]. The traditional way to treat these equalities is using the penalty method, transforming the constrained problem into an approximate unconstrained one [16], [6].

When calculating the fitness function, the penalization can be added to or multiplied by the objective function. There are several ways to do this, but the most common penalty type is the additive one [18]. If the constraints are not satisfied by some solution, the fitness value is changed proportionally to the violation, measured by some parameter μ. The larger the parameter, the closer the penalty problem is to the original one, but μ needs to be precisely chosen. A very large μ can lead to problems with ill-conditioning and too much emphasis on finding the feasible set. That way, the algorithm may end before expectation with sub-optimal results. If a very low one is picked, the algorithm may be clueless about how to reach the feasible region and return unfeasible solutions [2]. The major drawback of the penalty approaches is the requirement of a good penalty function definition and a proper tuning of penalty parameters, which can be challenging.

Even with the penalty approach application, the problem remains very complex. An improvement that can be done to EA's is the hybridization of the algorithm. By definition, something is hybrid when is powered by two or more sources. In many cases, the hybrid algorithm uses both the exploration advantage of EA's and the precision in local searches' algorithms. According to [8], there are many ways to hybridize an algorithm. Figure 1 shows some ways to do a hybridization, such as using other algorithms to develop better initial population, improve evolutionary operators or search the offspring of superior individuals. Hybridism is not necessarily better than the pure EA's, but it has to be as precise as them. So, it is worth testing if the time increase is not intolerable.

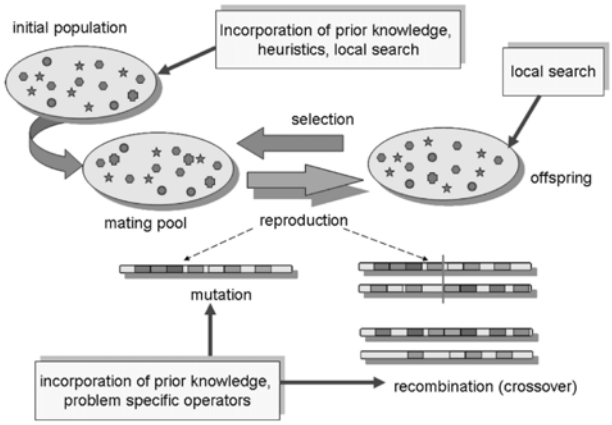

Figure 1. Hybridization possibilities [8].

In this paper, we propose a hybrid *Particle Swarm Optimization* (PSO) algorithm to solve some benchmark equality constraint problems. It is a very studied algorithm with lots of uses [10], but it has limitations, as described in [5]. Changing some aspects of the algorithm or hybridizing it can be done to overcome this limitations. Some previous works have tried to change how the particles moves (what we call topology) [3], and including a local search in it [4], both dealing with constrained problems. Our work maintained the original PSO almost entirely, only adding the inertia factor [13]. After generating a population of solutions with the EA, we enhance it with the *Iterated Local Search* (ILS) algorithm. The cost of extending the execution time is expected to be compensated by better function values at the end of the process.

The paper is organized as follows. In Section II, we explain the proposed ILS algorithm. Section III describes how to build the PSO and hybridize it with the local search algorithm. Section IV shows the results obtained through tests, and the conclusions are derived in Section V.

II. Iterated Local Search

The Iterated Local Search (ILS) algorithm is a simple, easy to implement, robust and highly effective metaheuristic

978-1-5090-6017-7/18/$31.00 ©2018 IEEE

[11]. It focuses on exploring a small region of the space of solutions, searching for local minima that could improve previously obtained points. The name comes from the fact that the algorithm repeatedly uses local search to find those improvements.

The method only requires one input, the starting point. Then the following steps occur:

1) The best point in a specified neighborhood is chosen, using a hill climbing method [12].
2) That point will be the center of the next neighborhood search.
3) The first two steps will be repeated until there is no upgrade for the current solution or if the maximum number of iterations is reached.

When the local search is done, it is expected to have found a local minimum. The next step is perturbing that point, and the result of that perturbation will be the start of the next search, exploring other neighborhood for an even better solution. Repeating that numerous times, the method expects to check attraction basins for the best minimum around the starting point. The algorithm ends returning the best point found over all the iterations. Figure 2 shows the procedure.

Each local search phase will be deterministic, returning always the same solution if it starts at the same point. However, the method is stochastic, since the perturbation is randomly chosen. This grants variability to the search, but requires a wise choice on the number of iterations: quick searches could be ineffective, while long ones could be a waste of time, possibly falling into the same attraction basins too many times.

As the reader can see, the basic structure of the strategy is very simple. An implementation is shown in Algorithm 1. The biggest possible problem should be the calibration of parameters. In this study, the following parameters were empirically used:

- The points examined by the Downhill function have only 1 variable different from the input. That variable has a $\pm 0.1\%$ of the variable range added to it. So, with n variables, n points will be compared with the input.
- If the Downhill doesn't make an improvement or the limit (set to 150 iterations) is reached, the local search is terminated.
- The next starting point is randomly put in the neighborhood correspondent to the hypercube with edges whose length is 2% of the variable range and is centered in the best-found point.
- The perturbation is repeated 100 times.

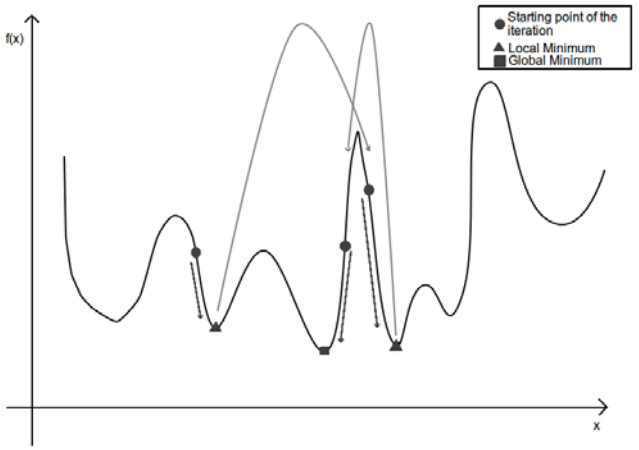

Figure 2. ILS graphic procedure

Algorithm 1 Iterated Local Search

```
procedure ILS(x_0)
    best ← x_0
    x_k ← x_0
    for k = 0 < maxiterations do
        for i = 0 < limit do
            function DOWNHILL(x_k)
                min ← x_k
                for all y ∈ V(x_k) do
                    if f(y) < f(min) then
                        min ← y
                    end if
                end for
                return min
            end function
        end for
        if f(min) < f(best) then
            best ← min
        end if
        x_k ← perturb(min)
    end for
end procedure
```

In this paper, the ILS will be used alongside with the PSO. The initial solution is going to be the best individual in the PSO's population in some predefined generations, and the point returned by the ILS algorithm will deterministically replace the worse individual of the population. This hybridism enhances the results obtained by the evolutionary algorithm by itself by exploring more accurately the areas occupied by the population.

III. PSO WITH ILS

The PSO algorithm, first introduced in 1995 by Eberhart and Kennedy [7], is a population-based cooperative algorithm, shares similar characteristics to the genetic algorithm, however, the manner in which the two algorithms traverse the search space is fundamentally different, to handle optimization problems. The version used here incorporates the mechanism of hybridization with a local search operator based in the iterated local search.

A. Real PSO

The PSO has two primary operators: velocity update and position update. Here, we use the global version of PSO, as follows:

1) The algorithm is initialized with an initial population, treated in the PSO as a set of particles, each particle is a possible solution to the problem, with random positions and velocities in the search space and fitness function computation for each particle.
2) Store the position of the best generation particle (called $gbest$) and the best position of each individual particle (called $pbest$).
3) Compare particle's fitness evaluation with particle's $pbest$. If the current value is better than $pbest$, then reset $pbest$ value equal to current value.
4) Compare population's fitness evaluation over previous best. If the current value is better than $gbest$, then reset $gbest$ value equal to current particle's index and value.
5) Update the velocity and position of the particle according to Equation 1 and Equation 2, respectively:

$$\begin{aligned} v_i^k(t) =& w^t * v_i^k(t-1) + c_1\gamma_{1i}(pbest_i^k - x_i^k(t-1)) \\ &+ c_2\gamma_{2i}(gbest_i^k - x_i^k(t-1)) \end{aligned} \tag{1}$$

wherein $v_i^k(t)$ is ith component of the velocity of the kth particle, $x_i^k(t)$ is the ith component of the position of the kth particle, in tth step of the algorithm; w is the inertia factor, which ranges from 0.9 to 0.4; c_1 and c_2 are the social parameters, which can be varied to make the particles have more tendency to go in the direction of $pbest$ and $gbest$, in this work both were used equally to 2,05; γ_{1i} and γ_{2i} are random values between 0 and 1.
From the velocity and the previous position of the particle its new position is calculated:

$$x_i^k(t) = x_i^k(t-1) + v_i^k(t) \tag{2}$$

6) Loop to item 2 until a criterion is satisfied.

In case of an individual is out of the allowed range, the reflection operator is executed to force the individual back into the feasible region. The reflection by the lower limit (x_L) and upper limit (x_U) is given respectively by

$$x_r = x_L + |x - x_L| \text{ or } x_r = x_U - |x_U - x|$$

where x is the individual outside of the feasible region and x_r represents the reflected individual. We use the chaotic random inertia weight strategy $w^t = -0.5 \times rand() + 0.9$.

B. Hybrid PSO

The hybrid PSO (real-PSO with ILS) algorithm, presented in this section, deals with nonlinear equality constraint using ILS method as a local search operator in the real-PSO algorithm. Some of the possibles reasons for hybridization are as follows [14]:

1) To improve the performance of the evolutionary algorithm (example: speed of convergence)
2) To improve the quality of the solutions obtained by the evolutionary algorithm
3) To incorporate the evolutionary algorithm as part of a larger system

This kind of approach is used by many global optimization procedures, and the main advantage obtained is that the search space of solutions is reduced to a "subspace" of local optimal. We show below the scheme of the Hybrid PSO algorithm:

Algorithm 2 Hybrid PSO

```
procedure
    Initialize PSO parameters
    Initialize PSO population
    if no stop criterion then
        Evaluate population
        Store pbest and gbest
        Update velocity and position
        Local search operator
    end if
end procedure
```

IV. Experiments and Results

In this section, we describe numerical experiments with a set of analytical problems in order to compare the PSO and PSO-Hybrid algorithms. The experiments were run using a 3.5 GHz Intel Core i7 CPU with 32 GB RAM and our implementation was done in MatLab. The Matlab language codes are available from the corresponding author.

The analytical tests were chosen of benchmark problem presented in [15] and [1]. The Problem 1 is a multi-modal known Rastrigin function, with only two variables and a simple quadratic restriction. The Problem 2 is harder since the objective function grows exponentially with every variable, and have three restrictions with three restrictions. Lastly, the Problem 3 have three variables and two restrictions, with a wider range, slightly harder than the first one. All of them are minimization problems, described as follows:

- **Problem 1:**

$$\begin{aligned} &\min f(x) = x'.A'.Ax - 10[1\ 1]\cos(2\pi Ax) \\ &\text{subject to: } (x_1-2)^2 + (x_2-2)^2 = 1 \\ &A = \begin{pmatrix} 1 & 0 \\ 0 & 4 \end{pmatrix} \\ &-4 \le x_i \le 4,\ i = 1,2 \end{aligned} \tag{3}$$

The optimum values for the functions is $f(x^*) = -1.3953$.

- **Problem 2:**

$$\begin{aligned} \min f(x) = \quad & e^{x_1x_2x_3x_4x_5} \\ \text{subject to:} \quad & \begin{cases} x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 - 10 = 0 \\ x_3x_2 - 5x_4x_5 = 0 \\ x_1^3 + x_2^3 + 1 = 0 \\ -2.3 \le x_i \le 2.3,\ i = 1,2; \\ -3.2 \le x_i \le 3.2,\ i = 3,4,5 \end{cases} \end{aligned} \tag{4}$$

The global optimum is $x^* = (-1.7171, 1.5957, 1.8272, -0.76364, -0.7636)$ wherein $f(x^*) = 0.0539498$.

- **Problem 3:**

$$\begin{aligned} \min f(x) = \quad & 1000 - x_1^2 - 2x_2^2 - x_3^2 - x_1x_2 - x_1x_3 \\ \text{subject to:} \quad & \begin{cases} x_1^2 + x_2^2 + x_3^2 - 25 = 0 \\ 8x_1 + 14x_2 + 7x_3 - 56 = 0 \\ 0 \leq x_i \leq 10, \ i = 1, 2, 3 \end{cases} \end{aligned} \tag{5}$$

The global optimum is at $x^* = (3.512, 0.217, 3.552)$ wherein $f(x^*) = 961.715$.

The $g(x) = 0$ type equality constraints were replaced by two inequality constraints of the form:

$$g_1(x) = -g(x) - \epsilon \leq 0$$

$$g_2(x) = g(x) - \epsilon \leq 0$$

with $\epsilon = 0.001$. If any inequality constraint g_i is violated then the objective function is penalty via

$$f = f + 10g_i(x).$$

Each algorithm was executed 30 times for each problem with the same parameters described in subsection III-A for both versions and

- particle population size = 100
- maximum number of generations = 200

. The local search operator was run every 5 generations.

The main purpose of this paper is the application of the proposed local search with a well-established PSO algorithm. Having that in mind, the effect of the parameter choices over the PSO performance is not addressed in this work. A more detailed and exhaustive parameter fine-tuning, including the local search frequency, will be carried out in a future work.

A. Results

The PSO-ILS hybridization was compared with the pure PSO and with another hybridization that uses a Broyden-Fletcher-Goldfarb-Shanno (BFGS) operator [9], in which an estimate of the inverse Hessian matrix is constructed iteratively, to define a search direction.

The maximum number of generation was the only stopping criterion for all algorithm versions. At the end of all algorithm runs, the mean convergence line, representing the mean value of the best individual throughout the 200 generations, is obtained for each problem test.

Figures 3, 4 and 5 show the convergence lines for each problem. In the graphs, the x-axes represent the generation and the y-axes represent the objective function value of the best individual.

Observing Figure 4 and 5, it is possible to see that the PSO-ILS has a higher speed of convergence compared with the PSO-BFGS and the pure PSO. Around the 15th generation, the PSO-ILS H found a solution which is a better value than the final solution found by the PSO-BFGS and pure PSO. For Problem 1, analyzing the convergence line given by Figure 3, the convergence speed of the PSO-ILS is slightly higher than that of the PSO-BFGS and pure PSO.

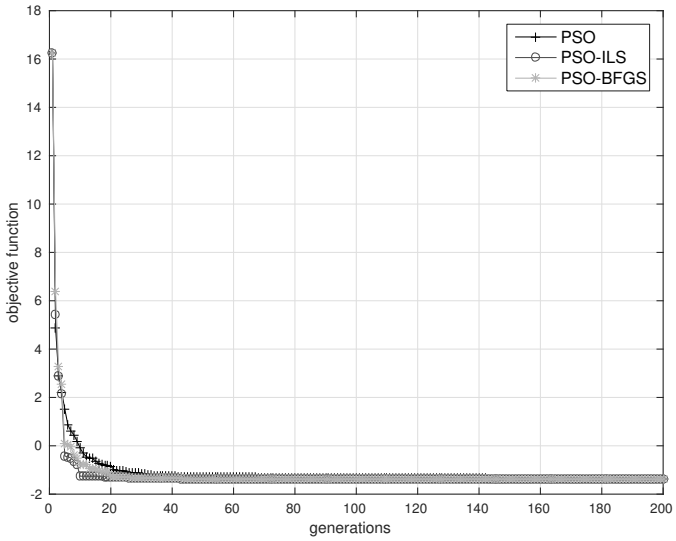

Figure 3. Convergence Lines for Problem 1.

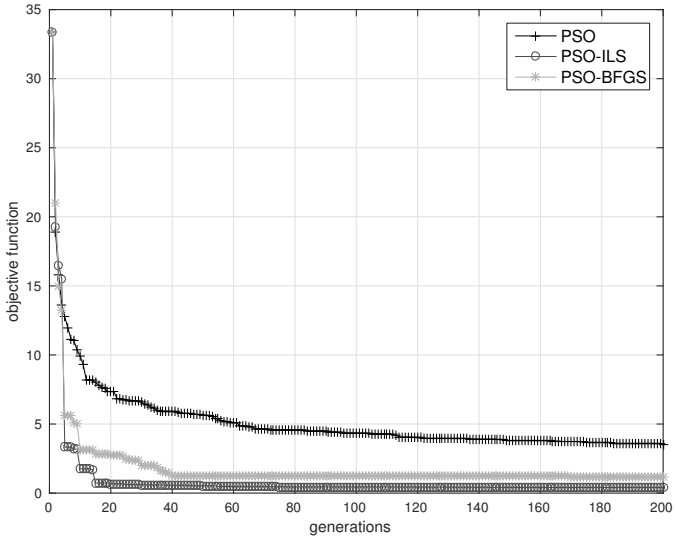

Figure 4. Convergence Lines for Problem 2.

With the goal of comparing the quality of final solution of the algorithm versions, a non-parametric statistical test is applied in a one versus all approach. The mean difference between the final solution of the algorithms, PSO-ILS versus PSO and PSO-ILS versus PSO-BFGS, is taken as a test statistic object. The significance of these observed results is assessed using a Monte Carlo simulation. The null hypothesis to be tested is stated as *There is no evidence that one of the algorithms is intrinsically better than the other one*. The central assumption of the Monte Carlo simulation is that, if the observed result has arisen by chance, then this value will not seem unusual in a distribution of results obtained through many random relabellings of the samples. For applying the statistical test, the following steps must be carried out:

1) Compute the object (the mean difference between the objective function values) for the samples for each algorithm. This value is called the observed value;
2) Reallocate randomly half of the samples to one

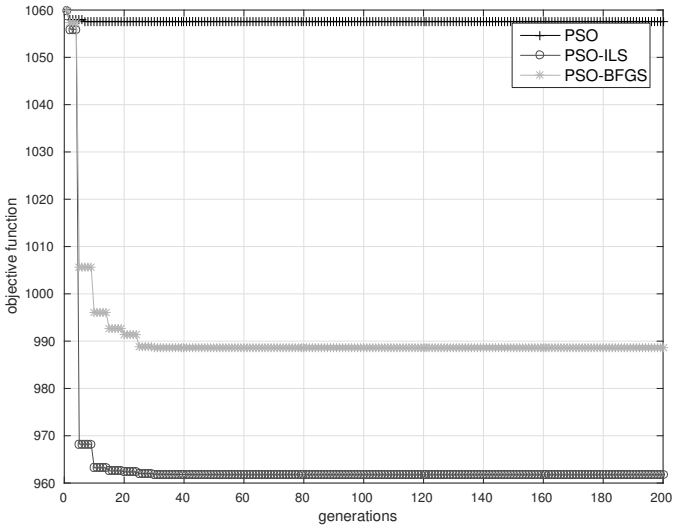

Figure 5. Convergence Lines for Problem 3.

algorithm and half to the other. Compute the object value as before;

3) Repeat the previous step until a large number of randomized object values have been generated and construct a distribution of these values;
4) If the observed value is not more extreme than a fraction of the resulting outcomes that corresponds to a significance level, then consider the null hypothesis; otherwise, reject it.

For Problem 1, after applying the statistical test, the null hypothesis can not be rejected meaning that there is no statistical difference between the final objective values for the pair of algorithms, (PSO-ILS, PSO) and (PSO-ILS, PSO-BFGS). However, the convergence speed of the PSO-ILS is clearly higher than the convergence speed of the other algorithms. Furthermore, a final solution can be found by PSO-ILS around the 15th generation.

Figures 6 and 7 show the statistical results using the final objective values of the algorithms considering Problem 2. Figures 8 and 9 show the statistical results using the final objective values of the algorithms considering Problem 3. The results are described by the gray histograms, while the observed result is depicted as a filled black circle over the same figures. The statistical tests have been performed using mean(PSO-ILS)-mean(·) and, therefore, positive differences (situated on the right of the histogram) favor PSO-ILS over the other algorithms.

The overall gain provided by the ILS hybridization is clear. Now we present the Table I with more precise information about runtime (best, worst, average and standard deviation) required for each algorithm.

In all the problems, the PSO-ILS took more time than the other algorithms, and that is expected since it checks so many perturbations, descending many times, each time evaluating points. It is, however, the most precise algorithm, making the hybridization advantageous if the biggest concern of the user is reducing the gap to the optimum. Moreover, the pure PSO and the BFGS hybridization lose accuracy when the problem

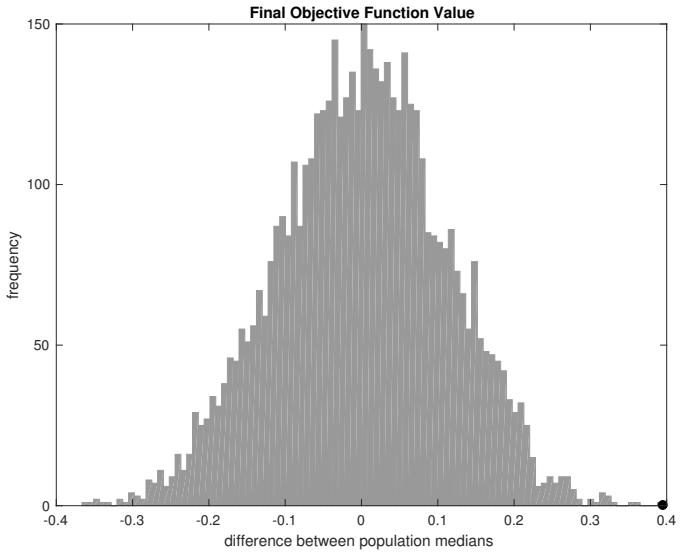

Figure 6. Statistical testing, using the final objective function values for Problem 2 considering PSO-ILS and PSO-BFGS. The result is statistically significant since the filled black circle lies on the right-hand side of the histogram.

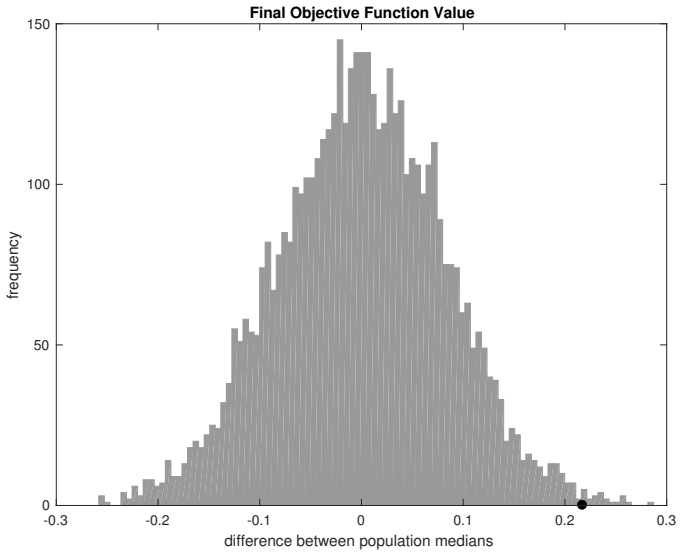

Figure 7. Statistical testing, using the final objective function values for Problem 2 PSO-ILS and PSO. The result is statistically significant since the filled black circle lies on the right-hand side of the histogram.

gets harder, while the ILS version keeps the precision, and it is the only reliable option for more complex problems with unknown optimum. The standard deviation is also lower in our approach, suggesting a stable algorithm, while the other algorithms can have some bad results depending on the initial seed of the random number generator.

V. Conclusion

When solving very complex optimization problems, it is desirable to get as close as possible to the best point, giving long time periods to the algorithms to evolve. Therefore, it is interesting to make a hybridization if the increased cost in execution time is compensated by doing the algorithm more

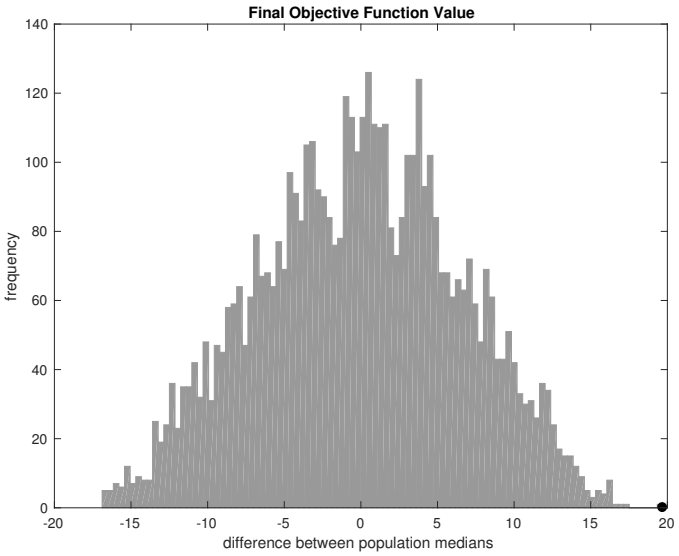

Figure 8. Statistical testing, using the final objective function values for Problem 3 considering PSO-ILS and PSO-BFGS. The result is statistically significant since the filled black circle lies on the right-hand side of the histogram.

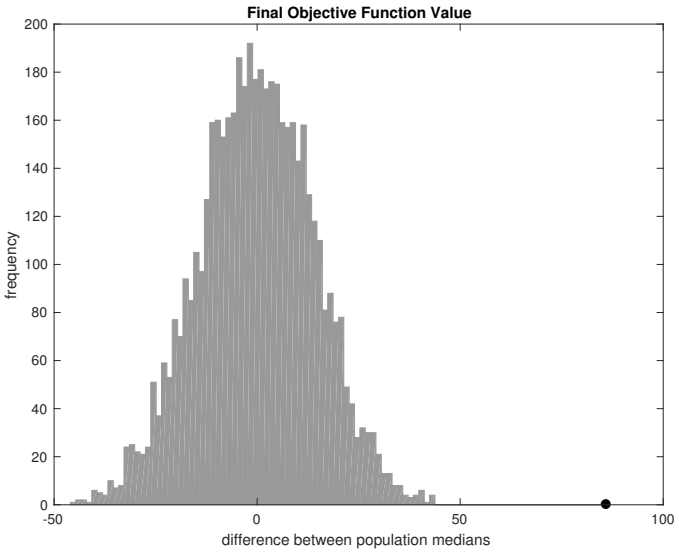

Figure 9. Statistical testing, using the final objective function values for Problem 3 PSO-ILS and PSO. The result is statistically significant since the filled black circle lies on the right-hand side of the histogram.

Table I. RUNTIME OF THE ALGORITHMS FOR EACH PROBLEM TEST.

Run time (s)	PSO			PSO-BFGS			PSO-ILS		
	P1	P2	P3	P1	P2	P3	P1	P2	P3
mean	0.53	0.37	0.37	1.27	0.94	0.91	4.39	14.66	3.25
std	0.02	0.01	0.05	0.05	0.03	0.06	0.09	12.77	0.14
min	0.50	0.35	0.30	1.22	0.89	0.84	4.27	7.85	3.07
max	0.62	0.41	0.49	1.42	1.00	1.08	4.69	61.09	3.61

robust. In this paper, we tested the performance of the PSO hybridization using the ILS algorithm in benchmark equality constrained problems. This type of problem has a very complex feasible region, so evolutionary algorithms need to attach other search strategies to be useful. The developed method could find solutions with tiny gaps to the optimum, and it has proved to be more accurate than another good hybridization, using the BFGS algorithm, and than the pure evolutionary algorithm.

ACKNOWLEDGMENTS

The authors thank UFOP and funding Brazilian agencies CNPq, Fapemig, and CAPES. We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan X Pascal GPU used for this research.

REFERENCES

[1] A. H. Aguirre, A. M. Zavala, E. V. Diharce, and S. B. Rionda, "Copso: Constrained optimization via pso algorithm," *Center for Research in Mathematics (CIMAT). Technical report No. I-07-04/22-02-2007*, 2007.

[2] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty, *Nonlinear programming: theory and algorithms*. John Wiley & Sons, 2013.

[3] M. R. Bonyadi, X. Li, and Z. Michalewicz, "A hybrid particle swarm with a time-adaptive topology for constrained optimization," *Swarm and Evolutionary Computation*, vol. 18, pp. 22–37, 2014.

[4] M. R. Bonyadi and Z. Michalewicz, "Locating potentially disjoint feasible regions of a search space with a particle swarm optimizer," in *Evolutionary constrained optimization*. Springer, 2015, pp. 205–230.

[5] ——, "Particle swarm optimization for single objective continuous space problems: a review," 2017.

[6] C. A. C. Coello, "Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art," *Computer methods in applied mechanics and engineering*, vol. 191, no. 11, pp. 1245–1287, 2002.

[7] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Micro Machine and Human Science, 1995. MHS'95., Proceedings of the Sixth International Symposium on*, 1995, pp. 39–43.

[8] C. Grosan and A. Abraham, "Hybrid evolutionary algorithms: methodologies, architectures, and reviews," in *Hybrid evolutionary algorithms*. Springer, 2007, pp. 1–17.

[9] D. G. Luenberger, *Linear and Nonlinear Programming*. Springer, 2003.

[10] R. Poli, "Analysis of the publications on the applications of particle swarm optimisation," *Journal of Artificial Evolution and Applications*, vol. 2008, 2008.

[11] H. Ramalhinho-Lourenço, O. C. Martin, T. Stützle *et al.*, "Iterated local search," Tech. Rep., 2000.

[12] B. Selman and C. P. Gomes, "Hill-climbing search," *Encyclopedia of Cognitive Science*, 2006.

[13] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," in *Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on*. IEEE, 1998, pp. 69–73.

[14] P. Somasundaram, R. Lakshmiramanan, and K. Kuppusamy, "Hybrid algorithm based on ep and lp for security constrained economic dispatch problem," *Electric Power Systems Research*, vol. 76, no. 1, pp. 77–85, 2005.

[15] E. F. Wanner, F. G. Guimaraes, R. R. Saldanha, R. H. C. Takahashi, and P. J. Fleming, "Constraint quadratic approximation operator for treating equality constraints with genetic algorithms," in *2005 IEEE Congress on Evolutionary Computation*, vol. 3, 2005, pp. 2255–2262 Vol. 3.

[16] E. Wanner, "Operadores para algoritmos genéticos baseados em aproximações quadráticas de funções de variáveis contínuas," Doctoral thesis, Universidade Federal de Minas Gerais, 2006.

[17] E. F. Wanner, F. G. Guimaraes, R. H. Takahashi, and P. J. Fleming, "Local search with quadratic approximation in genetic algorithms for expensive optimization problems," in *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*. IEEE, 2007, pp. 677–683.

[18] Ö. Yeniay, "Penalty function methods for constrained optimization with genetic algorithms," *Mathematical and Computational Applications*, vol. 10, no. 1, pp. 45–56, 2005.