An Efficient Method to Obtain Bifurcation Diagrams based on PSO Algorithms

Bryan E. Martínez, Monserrat A. Castro-Coria, Jaime Cerda and Alberto Avalos

***Abstract*—A bifurcation diagram is used to analyze the dynamics of a system based on its parameters, and when these are allowed to change. They provide information about how the roots of a dynamic system change as those parameters change. During this analysis you get information that helps maintain the stability of a system. It seems to be an easy task, which is not because throughout the process of changing parameters, some roots enter the real domain that seems as if they appear and others enter the complex domain which could be taken as if they were disappearing . This paper proposes a methodology to address the generation of bifurcation diagrams. An alternative to this approach is to use multimodal evolutionary optimization algorithms, many of which are based on NichePSO. This paper presents a niche algorithm based on PSO called NichePSO stochastic but works mainly by following the individual trajectories of the solutions. The results allow us to affirm, at least for univariate problems, that it is more efficient than those that only use NichePSO.**

***Index Terms*—PSO, NichePSO, Multiobjective Optimization, Parametric Optimization, Bifurcation Diagram, Bisection.**

I. Introduction

BIFURCATION Diagrams are used to analyze the dynamics of a system when its parameters are allowed to change. They provide information about how the roots of a dynamic equation change as those parameters change. Even that they seems to be an easy task, this is not the case since, along the parameter changing process, some roots get into the real domain that seems like if they appeared and others get into the complex domain that we take as if they were disappearing. In this work a methodology is proposed to approach the generation of the bifurcation diagram. This is equivalent to solve multimodal optimization problems that are faced using global optimization techniques. When faced with the formal mathematical tools to obtain these bifurcation diagrams most of the methods proposed require initial values that imply a strong knowledge of the system to be analyzed and in general they are not very fast. An alternative to this approach is to use global multimodal evolutionary optimization algorithms, one of that is NichePSO.

NichePSO algorithm was developed by Bris et al. [1] based on PSO to find multiple solutions to multimodal problems. It was the first PSO niching technique that introduced a novel PSO algorithm to detect multiple optimal in a multimodal problem. NichePSO begins with a swarm known as the main swarm, which contains all the particles. Then, as a particle converge into a possible solution, a sub-swarm is created by grouping the particles that are closer to the possible solution. These are removed from the main swarm and continue within their sub-swarm to refine and maintain such solution. Subsequently, the main swarm contracts as new sub-swarms are generated. It is considered that NichePSO has converged when the sub-swarms no longer improve the solutions that they represent. The best overall position of each sub-swarm is then taken as one of the optimal solutions [2]–[5].

NichePSO has been used to optimize the multi well placement in oil field development with the objective to maximize the cumulative production [6], others applications are ensembles of bi-clusters generated by NichePSO [7], visual tracking based on NichePSO [8] and Bifurcation Diagram Tool (BDT) based on NichePSO, a meta-heuristic capable of optimizing functions [9]. The last work has used this strategy using only NichePSO along the limit of variation of parameter but they do not take into account previous solutions. This paper presents a NichePSO-based algorithm which stochastically call NichePSO but it works mainly following individual trajectories of the solutions.

The rest of the paper is as follows: Section II briefly describes PSO and NichePSO algorithms. Then, a short description of bifurcation diagrams is given in section III. After this, the proposed methodology is described in Section IV. Experimental results are presented and discussed in Section V. Finally, conclusions and future work are presented in Section VI.

II. PSO and NichePSO

The proposed algorithm relies heavily on PSO and NichePSO. The first is used with a modification when the flock is initialized, so the actual solutions can be found using previous solutions, the second will be called stochastically in order to assure we always have the right number of solution. Therefore, in this section, the PSO as well as the NichePSO algorithms basic concepts are described.

A. PSO

The PSO algorithm has ben applied a number of problems i.e. solve permutation problems and training neural networks. The principle of PSO is find the maximum or minimum objective for problems with only a solution while NichePSO finds the maxima and minima objectives to solve problems with multiple solutions. NichePSO begins with a swarm known as the main swarm, which contains all the particles. As soon as a particle converge into a possible solution

Manuscript received July 23, 2018; revised July 31, 2018. This work was supported in part by the Facultad de Ingenieria Electrica at the Universidad Michoacana de San Nicolas de Hidalgo.

Bryan E. Martínez is with the Facultad de Ingenieria Electrica at the Universidad Michoacana de San Nicolas de Hidalgo, Morelia, Michoacan, Mexico (e-mail: emartinez@dep.fie.umich.mx).

Monserrat A. Castro-Coria is with the Division de Estudios de Posgrado at the Universidad Michoacana de San Nicolas de Hidalgo, Morelia, Michoacan, Mexico (e-mail: monserrat@dep.fie.umich.mx).

Jaime Cerda is with the Facultad de Ingenieria Electrica at the Universidad Michoacana de San Nicolas de Hidalgo, Morelia, Michoacan, Mexico (e-mail: jcerda@umich.mx.)

Alberto Avalos is with the Facultad de Ingenieria Electrica at the Universidad Michoacana de San Nicolas de Hidalgo, Morelia, Michoacan, Mexico (e-mail: javalos@umich.mx).

ISBN: 978-988-14048-1-7
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

a sub-swarm is created by grouping the particles that are closer to the best solution. The basic PSO algorithm was developed to find unique solutions to optimization problems, some variations of PSO are to solve problems such as multi objective optimization, dynamically changing objective functions and location of multiple solutions.

A variation of PSO are techniques of niches. The NichePSO algorithm was developed by Bris et al. [1]. It employs PSO to find multiple solutions to multimodal problems. It was the first PSO niching technique that introduces a novel PSO algorithm to detect multiple optimal in a multimodal problem whose implementation can be parallelized. The particle swarm concept was to simulate the graceful of choreography of a bird flock, that govern the ability to fly synchronously and suddenly chance direction or velocity. The position of particles are based on the social psychological tendency of individuals to emulate other individuals, and is given by

$$x_i(t+1) = x_i(t) + v_i(t+1) \tag{1}$$

where $v_i(t)$ denotes the velocity vector or the i-th particle. The velocity update has the form:

$$v_{ij}(t+1) = v_{ij}(t) + c_1 r_{1j}(t) p_{ij}(t) + c_2 r_{2j}(t) s_{ij}(t) \tag{2}$$

where the subindex ij denotes de j-th entry corresponding to i-th particle, and $p(t)$ is the cognitive component and $s(t)$ the social component, c_1 and c_2 are positive acceleration constants used to the contribution of the cognitive and social components, r_{1j} and r_{2j} are random values from a uniform distribution $U[0,1]$. Each step t a particle i updates its velocity and position, were the new velocity $v_i(t+1)$, is the sume of three terms, the previous velocity $v_i(t)$, to the distance from $lbest$ and the best position particle from $gbest$.

The PSO is in Algorithm 1 where the first step for PSO algorithm is the initialization of the main swarm, where cognitive component, social component, position and velocity need to be specified, in line 3 the fitness function is evaluated for each particle. In line 11 and 12 the position and velocity are updated; this process is repeated until a stopping condition is satisfied.

Algorithm 1 PSO Algorithm

```
1: Create and initialize an n_x-dimensional swarm;
2: repeat
3:    for each particle i = 1, ..., n_s do
4:       if f(x_i) < f(y_i) then
5:          y_i = x_i;
6:       end if
7:       if f(y_i) < f(ŷ_i) then
8:          ŷ = y_i
9:       end if
10:   end for
11:   for each particle i = 1, ..., n_s do
12:      update the position using equation 1;
13:      update the velocity using equation 2;
14:   end for
15: until stopping condition is true;
```

B. NichePSO

A PSO based algorithm called NichePSO assumes as an objective to find the different maxima of the problem. NichePSO updates the best particle position and velocity while also using PSO for the rest of the particles. The updating formulae are as follows

$$x_{rj}(t+1) = y_j(t) + w v_{rj}(t) + p(t)(1 - 2r_2(t)) \tag{3}$$

$$v_{rj}(t+1) = -x_{rj}(t) + y_j(t) + w v_{rj}(t) + p(t)(1 - 2r_2(t)) \tag{4}$$

The NichePSO algorithm is the one presented in Algorithm 2.

Algorithm 2 NichePSO Algorithm

```
1: Create and initialize a n_x-dimensional main swarm, S;
2: repeat
3:    Train the main swarm, S, for one iteration using the
      cognition-only model;
4:    Update the fitness of each main swarm particle, S.x_i;
5:    for each sub-swarm S_k do
6:       Train sub-swarm particles, s_k.x_i using a full model
         PSO;
7:       Update each particle's fitness;
8:       Update the swarm radius S_k.R;
9:    end for
10:   If possible, merge sub-swarms;
11:   Allow sub-swarms to absorb any particles from the
      main swarm that moved into the sub-swarm;
12:   If possible, create new sub-swarms;
13: until stopping condition is true
14: return S_k.ŷ for each sub-swarm S_k as a solution;
```

The initialization of the main swarm is done in line 1. In line 3 this swarm is trained with the cognitive model only. In line 4 the fitness function for each particle is updated, then in line 6 the sub-swarms are trained using a PSO full model, after updating the fitness for each particle in line 7, in line 8 the swarms radius are update. In line 10, the sub-swarms are merged if is possible. Finally the best particle of each sub-swarm is returned as a solution in line 14.

The NichePSO has good performance when finding the solutions to multimodal problems. However, it was found that the current sub-swarm merging and particle absorption strategies are premature, and limits exploration in the main swarm. That is shown in [10] where they describe original merging and absorption routines within the NichePSO. This can be enhanced by including behavioral information in the decision making process whether to merge or not. They proposes four new strategies, two for enhanced merging (Directional Based Merging and Scatter Merging) and two for enhanced absorption (Directional Based Absorption and Euclidean Diversity Absorption).

III. Bifurcation Diagrams

A bifurcation refers to a qualitative change in the behavior of a dynamical system as some parameter on which the system depends varies continuously [11]. A point (x_0, θ_0) is a bifurcation point of equilibrium for $x_t = f(x;\theta)$ if the

ISBN: 978-988-14048-1-7
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

number of solutions of the equation $f(x;\theta) = 0$ for x in every neighborhood of (x_0, θ_0) is not a constant independient of θ.

A Bifurcation Diagram (BD) is the generation of a graph which explicitly tell us about the dynamics of the equation roots as its parameters are continuously changing. Each of this points can then be evaluated for their stability assessment, for which there exist several well stablished methods and therefore are left out of this work as does not add anything new. An example of a one dimensional BD [9] is the equation 5:

$$\dot{x} = \theta x + x^3 - x^5 \tag{5}$$

has a subcritical pitchfork bifurcation at $(x, \theta) = (0, 0)$. When solution $x = 0$ loses stability as θ passes through zero, the system can jump to one of the distant stable equilibria corresponding to $x = \pm 1$ at $\theta = 0$.

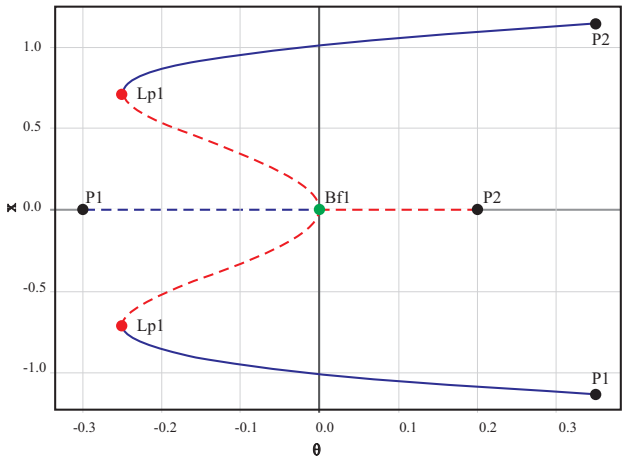


Fig. 1. Bifurcation Diagram $x_t = \theta x + x^3 - x^5$

Figure 1 represents a BD where θ is the parameter which will be changing and therefore the equation roots will be probably also changing. Here, we can observe that when $\theta = -0.25$ a two bifurcation points were found one at $x = -0.7$ and the other at $x = 0.7$, moreover, these two bifurcations appear (or disappear if we were diminishing θ), please do notice we now have three roots, instead of one which we had if we go back an infinitesimal distance from $\theta = -0.25$. We can also observe that if we increase by an infinitesimal value θ, we will have five roots. Another interesting point is when $\theta = 0$ where before reaching that value we had five roots and in that value three roots merge to just one root e.g $x = 0$. $Bf1$ point is the bifurcation, $LP1$ are turning points or limit points, $P1$, $P2$ and $LP1$ are branch points.

IV. Proposed Methodology

Torres et al. [9] describe the normal NichePSO algorithm is constantly applied to build the BD but the previous solution to the problem is ignored, which makes it a slow process. The methodology we propose takes the previous solution as a starting point to find the following solution. This poses two additional problems which can arise along the bifurcation diagram generation process: Appearance of new roots and disappearing of existing roots. To this end, we are proposing new heuristics in order to deal with them. In the following section we will describe the main concepts of our methodology.

A. Problem Transformation

PSO as well as NichePSO were designed to deal with optimization problems, however in order to find the bifurcation diagram we need to find the zeroes of the system. To this end, we need to transform our dynamic system function f into a function g where all the zeroes will be the maximal points of the transformed function g. Specifically, g is defined as equation 6.

$$g(x) = \frac{1}{1 + \|f(x)\|} \tag{6}$$

This function will be the one we will use to evaluate the fitness of the elements of the swarm.

B. Bisections Method

Traditionally, the bisection method [12] has been used as a means to find the root of a given function. This method iteratively converges on a point where $f(x) = 0$ which is taken as the solution of the process. It basically changes a reference point, x, at the middle of the space search i.e. $[x_a..x_b]$ and the deciding which of the halves to explore i.e. $[x_a, x]$ or $[x, x_b]$ based on some criteria. Usually this criterion is the sign of the function evaluated at x, x_a, x_b. In our method our criterion will be the number of roots at x, x_a, x_b. In this contribution we will use this method as a mean to find the bifurcation point. This process will be called when, as a result of the NichePSO application, a different number of roots, from those we were processing, are found.

The bisection method in our case is illustrated in Figure 2 where we can see, when NichePSO is applied, a different number of roots at the ends of the green range. This fact will fire the bisection process which moves the reference point at the middle of the green range and now decides to search the blue range as it is there where a different in the number of roots are present at the limits of this range. This process will go on until, if x_n is the bifurcation point, $|x_n - x_{n-1}| < \epsilon$.

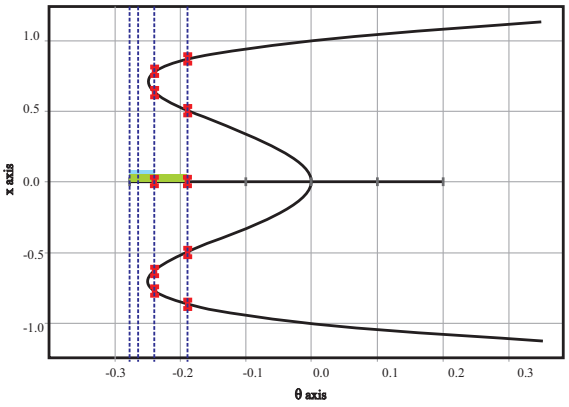


Fig. 2. Bisection Method

C. The fastBDPSO algorithm

The DE NichePSO Algorithm starts with the NichePSO algorithm and exploring with bisection method to obtain the roots and new points, then the diagram was created obtained by particles tracking and gets the slope. Then call Differential Evolution algorithm for convergence the niches; last use the history for search the next particles as show in Algorithm 3.

ISBN: 978-988-14048-1-7
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

The main algorithm proposed in this work is shown in algorithm 3. Line 8 calls NichePSO which returns a set of solutions which are stored in X. Then we will repeat the following process. If stochastically determined, apply NichePSO() (line 12) and if the number of solutions are greater from those in X, the bisection method is applied to find the bifurcation point (line 15). NichePSO is called again after that point $\theta + \delta\theta$ (line 18). From that point we will apply the trackerPSO algorithm to follow the track of each solution from θ_x to θ_y (lines 19-22). Please do notice that in this tracking process NichePSO will not be called again. Line 26 will call trackerPSO to follow the solutions path and finally in line 27 θ is increased. This process will be repeated until $\theta > \theta_f$.

Algorithm 3 fastDBPSO

1: # X, Y, sets of solutions
2: # θ_o, θ initial value
3: # θ_f, θ final value
4: # $\delta\theta$, θ increment
5: # τ, NichePSO application probability
6: # r, random number from a uniform distribution
7: $\theta \leftarrow \theta_o$;
8: $X \leftarrow NichePSO(\theta)$;
9: $\theta_x \leftarrow \theta$;
10: **repeat**
11: **if** $r < \tau$ **then**
12: $Y \leftarrow NichePSO(\theta)$;
13: **if** $|X| < |Y|$ **then**
14: $\theta_y \leftarrow \theta$;
15: $\theta \leftarrow bisection(\theta_x, \theta_y) + \delta\theta$;
16: $X \leftarrow Y$;
17: $\theta_x \leftarrow \theta_y$;
18: $Y \leftarrow NichePSO(\theta)$;
19: **for** $\theta = \theta_y \ldots \theta_x$ **do**
20: $Y \leftarrow trackerPSO(Y)$;
21: $\theta \leftarrow \theta + \delta\theta$;
22: **end for**
23: $X \leftarrow Y$;
24: **end if**
25: **end if**
26: $X \leftarrow trackerPSO(X)$;
27: $\theta = \theta + \delta\theta$;
28: **until** $\theta > \theta_f$

On the other hand the trackerPSO algorithm is described by algorithm 4, This heavily relies on PSO, the only difference being in the initialization part where the best particle becomes the previous solution of that path and a five particles swarm is created around such particle (line 3). This will be done for every particle of the solution passed to trackerPSO (X). It is worthy to note that in this case the social component will be highly more important that the cognitive component therefore in the PSO algorithm the following must be fulfilled $C_1 << C_2$.

V. Experimental Results

NichePSO is computationally expensive, for this reason this is only employed in case it is needed and we heavily rely on tracking previous solutions using simple methods

Algorithm 4 trackerPSO(X)

1: # X, sets of solutions
2: **for** each $x \in X$ **do**
3: $x_s \leftarrow$ Create a five particles biased swarm around x;
4: $y \leftarrow PSO(x_s)$;
5: mergeSolutions();
6: **end for**

i.e. PSO and DE. The main difference is that we use fewer particles to follow such solutions as the previous solution has to be very close to the one we are looking for. Therefore, the efficiency as well as the precision is improved. This section show some experiments in relation with time response as well as accuracy for several systems. The experiments were conducted on an laptop computer MacBook Air, 4 Gb RAM, 1600 MHz DDR3, 1.6 GHz Intel Core i5, using macOS Sierra version 10.12.5. To render the results a bifurcation diagram plotting tool, called BDT (Bifurcation Diagram Tool), was implemented. In these experiments some parameters were measured, these are: time for creating the bifurcation diagram, precision to find the roots, the minimum number of particles necessary for good performance, apply NichePSO (BD-NichePSO), and track with PSO (Fast BD-PSO) as well as DE (FAST BD-DE).

The first column of Table I shows all the used parameters as well as nd a determinate iterations number for each test.

TABLE I
Niche Parameters Setting

Parameters	Niche	Fast BD-PSO	Fast BD-DE
Main Swarm Size	40, 65, 130 & 260	5	5
Learning Factor C_1	1	0.8	-
Learning Factor C_2	2	2	-
Inertia weight W	0.5	0.5	0.4
Cross Rate	-	-	0.8
Iteration number	65	30	30

With the purpose of comparing the efficiency a modified version of fastBDPSO implemented , where instead of using PSO in the tracking algorithm (line 4, algorithm 4) we use differential evolution which will be called Fast BD-DE. We have as reference the algorithm which uses NichePSO along all the different values of θ, which will be called simpleNichePSO. We also test the cases for $\delta\theta = 0.1, 0.01, 0.001$ steps for the three algorithms i.e. simpleNichePSO, fastBD-PSO and fastBD-DE algorithms using the same equation in a one dimensional diagram. Figure 3 shows the results of the different methods when applied to function $f(\dot{x}, \theta) = \theta x + x^3 - x^5$. All the cases were able to generate the BD, but as later will be shown, fastBDPSO outperforms the other two algorithms.

To realize the tracking, the strategy used was to follow the roots at point where these are merged using bisections, and from that point rebuilt the bifurcation diagram up to the point where simpleNichePSO fired the application of bisections. The points we are tracking are used predict the new positions as show Figure 4, then the history of particles is used to direction indicator. The idea of tracking employing Fast BD-PSO algorithm is to ensure particle rejection among them using a threshold from the best position found before.

ISBN: 978-988-14048-1-7
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

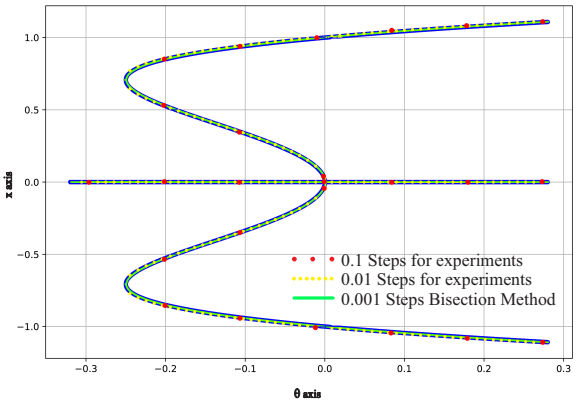
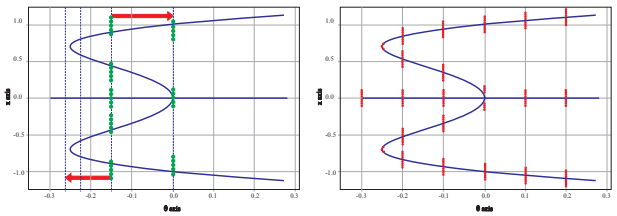
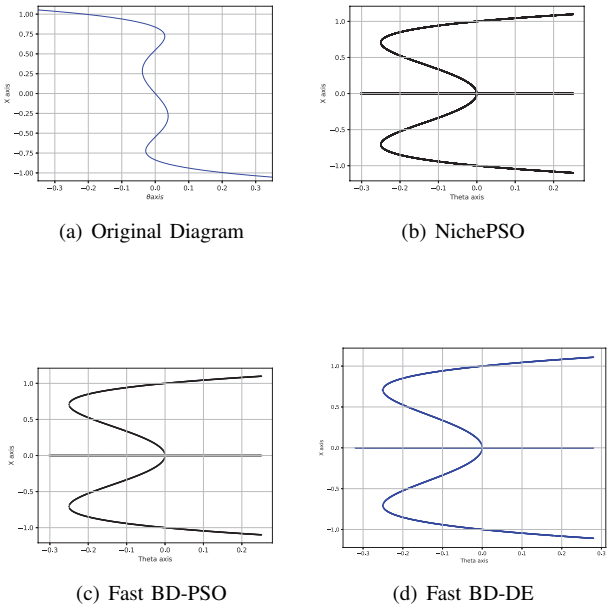


Fig. 3. Test with $f(\dot{x}, \theta) = \theta x + x^3 - x^5$



(a) 5 Particles (b) 8 Particles

Fig. 4. Particle Tracking with 5 and 8 particles.



(a) Original Diagram (b) NichePSO

(c) Fast BD-PSO (d) Fast BD-DE

Fig. 5. Results for $-x^5 + x^3 + x\theta$ Function

Table II shows some tests performed for all functions shown in first column. The second column show the value ranges for the functions. The third column shows the step size, and the last columns show the time required buy simpleNichePSO to obtain the BD with different number of particles i.e. 40, 65, 130 and 260. Elements in red were not able to generate the BD, therefore a swarm of 130 particles was decided to be used for simpleNichePSO.

TABLE II
NICHEPSO FOR DIFFERENT NUMBER OF PARTICLES

Function	Ranges	Step	Niche			
			40	65	130	260
$-x^5 + x^3 + x\theta$	$x = (-1.4, 1.4)$	0.1	0.6722	0.9336	1.6120	2.8532
	$\theta = (-0.32, 0.25)$	0.01	5.6544	7.6904	13.2951	23.5480
		0.001	54.3783	76.2523	125.6771	219.5866
$4x - x^3 + \theta$	$x = (-4, 4)$	1	1.4843	1.7264	2.5844	4.54245
	$\theta = (-4, 4)$	0.1	11.9331	14.7582	22.0439	36.9021
		0.01	115.3369	141.7751	210.6229	334.1761
$x + \theta - x^2$	$x = (-10, 10)$	1	6.0179	6.4065	8.3627	12.0327
	$\theta = (-10, 10)$	0.1	54.0352	58.5006	77.8289	115.5978
		0.01	506.0416	546.2896	712.4901	1034.6116

Figure 5 shows the results of the experiments using the functions and methods there described.

TABLE III
PARTICLES TRACKING TIME

Function	Step	Fast BD-PSO	Fast BD-DE	Niche
$-x^5 + x^3 + x\theta$	0.1	8.2676	17.7030	1.6120
	0.01	9.0056	20.443915	13.2951
	0.001	22.7764	52.292652	125.6771
$4x - x^3 + \theta$	1	76.2738	151.0828	2.5844
	0.1	79.3385	155.2536	22.0439
	0.01	81.1868	168.5666	210.6229
$x + \theta - x^2$	1	77.3132	160.3031	8.3629
	0.1	70.7938	158.2267	77.8289
	0.01	78.4142	174.9785	712.4901

As it can be seen, there is a Difference of time in tracking between Fast BD-PSO and Fast BD-DE. This was related to the heuristic of the PSO algorithm which is based on repulsion between particles with umbral to generate all the candidates around a point. Therefore, it is forced to converge faster using best position which is the solution to that path in the previous iteration. Tracking with Fast BD-DE algorithm the basic idea is the same to the previous case, but now mutation functions were used as well as the recombination that characterizes this algorithm. Consequently more iterations are needed to converge directly affecting the performance. The tracking of the particles is shown in Table III. As we can see the computational time for fastBD-PSO is half the time used by fastBD-DE in all cases and in general it outperforms simpleNichePSO.

VI. CONCLUSIONS AND FUTURE WORK

In this contribution an improved method based on PSO algorithms has been presented. Previous works have used such strategy using only NichePSO along the parameter variation limit but they do not take into account previous solutions. The use of bisections to speed the process to find the bifurcation point has been a efficient tool to this end. The results allow us to claim, at least for univariate problems, it is more efficient than those which only use NichePSO. The ideal solution would be to detect immediately the bifurcation point but it would require $\delta\theta \approx 0$ and therefore would be very time consuming Overall, when compared with the simpleNichePSO as well as fastBD-DE algorithms we have obtained All this work have been implemented in the one-dimensional space. Some other strategies must been developed in order to extend this work to higher dimensions. Furthermore, the algorithms to track the individual solutions can be mapped directly into a parallel implementation which well speed up the process.

ISBN: 978-988-14048-1-7
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

REFERENCES

[1] R. Brits, A. P. Engelbrecht, and F. Van den Bergh, "A niching particle swarm optimizer," in *Proceedings of the 4th Asia-Pacific conference on simulated evolution and learning*, vol. 2. Singapore: Orchid Country Club, 2002, pp. 692–696.

[2] B.-Y. Qu, P. N. Suganthan, and S. Das, "A distance-based locally informed particle swarm model for multimodal optimization," *IEEE Transactions on Evolutionary Computation*, vol. 17, no. 3, pp. 387–402, 2013.

[3] S. Bird and X. Li, "Adaptively choosing niching parameters in a pso," in *Proceedings of the 8th annual conference on Genetic and evolutionary computation*. ACM, 2006, pp. 3–10.

[4] I. Schoeman and A. Engelbrecht, "Using vector operations to identify niches for particle swarm optimization," in *Cybernetics and Intelligent Systems, 2004 IEEE Conference on*, vol. 1. IEEE, 2004, pp. 361–366.

[5] X. Li, "Adaptively choosing neighbourhood bests using species in a particle swarm optimizer for multimodal function optimization," in *Genetic and Evolutionary Computation Conference*. Springer, 2004, pp. 105–116.

[6] G. Cheng, Y. An, Z. Wang, and K. Zhu, "Oil well placement optimization using niche particle swarm optimization," in *Computational Intelligence and Security (CIS), 2012 Eighth International Conference on*. IEEE, 2012, pp. 61–64.

[7] L. Menezes and A. L. Coelho, "On ensembles of biclusters generated by nichepso," in *Evolutionary Computation (CEC), 2011 IEEE Congress on*. IEEE, 2011, pp. 601–607.

[8] H. T. Yao, H. Q. Chen, and T. F. Qin, "Niche pso particle filter with particles fusion for target tracking," in *Applied Mechanics and Materials*, vol. 239. Trans Tech Publ, 2013, pp. 1368–1372.

[9] O. V. Torres, J. C. Jacobo, and J. J. Flores, "A bifurcation diagram tool based on nichepso," in *Power, Electronics and Computing (ROPEC), 2013 IEEE International Autumn Meeting on*. IEEE, 2013, pp. 1–5.

[10] A. P. Engelbrecht and L. Van Loggerenberg, "Enhancing the nichepso," in *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*. IEEE, 2007, pp. 2297–2302.

[11] T. Zhou, *Bifurcation*. New York, NY: Springer New York, 2013, pp. 79–86. [Online]. Available: https://doi.org/10.1007/978-1-4419-9863-7_500

[12] E. W. Weisstein, "Bisection. From MathWorld-A Wolfram Web Resource," http://mathworld.wolfram.com/Bisection.html, 2017. [Online]. Available: [Online;accessed08-August-2017]

ISBN: 978-988-14048-1-7
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)