Parallel Reverse Niching PSO for Multimodal Optimization

Elnaz Zafarani-Moattar, Hamid Haj-Seyed-Javadi, and Mohammad-Reza Feizi-Derakhshi

***Abstract*—**Nowadays multimodal optimization is one of the issues in optimization problems. Most of existing works are done on GA and PSO. They try to change these algorithms to give them ability of finding all optima's. Niching is well known-method for this purpose. The existing methods employ Euclidian distance metric to calculate similarity of particles to prevent them from converging to single optima. Finding neighborhood by Euclidian distance is a time consuming process. Therefore, a new method, named Parallel Reverse Niching, is proposed in this paper. In this method, swarm of particles is divided into many subswarm or populations which work together in parallel. Reverse Niching is done by penalty function. Penalty function uses global best of other populations as center of Gaussian functions. Proposed method is also a parallel algorithm which uses capability of parallel processing to speed-up. Proposed method have been tested on five widely used test functions and experimental results show improvement in compare with some other existing algorithms.

***Keywords*—** Multimodal- Niching- Particle Swarm Optimization, Parallel.

I. Introduction

NOWADAYS multimodal optimization is one of the issues in optimization problems. Multimodal optimization refers to locating multiple peaks/optima in the search space in a single run. So, the goal is to find a set of results, not just a single one. In this paper our goal is to find a set of global/local results which will be done in parallel. Most of previous works, try to change existing algorithms in order to be able to find all optima. Most of previous works were done on Genetic Algorithm and PSO. In this paper the PSO algorithm will be changed in order to give it multimodal ability. PSO is a population based algorithm. In the proposed algorithm, swarm is divided into several populations which work in parallel. The only thing that is exchanged, is their global best values. The goal is to not only use parallelization features but also offer a multimodal optimization. The idea for multimodal optimization is reverse niching. Multiple parallel populations with a Gaussian penalty function are used to prevent particles from converging to a single optimum. This function's role is to prevent other population's particles to get close to this population's global best.

Elnaz Zafarani-Moattar is with the Department of Computer Engineering, Tabriz Branch, Islamic Azad University, Tabriz, Iran. (corresponding author's phone: +984113396117; e-mail: e.zafarani@iaut.ac.ir).

Hamid Haj-Seyed-Javadi is with the Department of Mathematics and computer science, Shahed University, Tehran, Iran. (e-mail: h.s.javadi@shahed.ac.ir).

Mohammad-Reza Feizi-Derakhshi is with the Department of Computer Engineering, University of Tabriz, Tabriz, Iran. (e-mail: mfeizi@tabrizu.ac.ir).

II. Related Works

PSO algorithm has been regarded as a powerful means to solve for complex optimization problems. This computational paradigm was introduced by Kennedy and Eberhart [1] and its model is directly related to the organization and behavior of animals which live in groups such as flock of birds or swarm of insects.

A parallel version of the particle swarm optimization (PPSO) algorithm is presented together with three communication strategies by Shu-Chuan Chu and et.al [2] which can be used according to the independence of the data. They used three strategies. The first strategy is designed for the parameters of solutions that are independent or are only loosely correlated such as the Rosenbrock and Rastrigrin functions. The second communication strategy can be applied to those parameters that are more strongly correlated such as the Griewank function. In cases where the properties of the parameters are unknown, a third hybrid communication strategy can be used. Experimental results demonstrate the usefulness of the proposed PPSO algorithm.

Particle swarm optimization (PSO) algorithm is a population-based algorithm for finding the optimal solution. Because of its simplicity in implementation and fewer adjustable parameters compared to the other global optimization algorithms, PSO is gaining attention in solving complex and large scale problems. However, PSO often requires long execution time to solve those problems. Bo Li and Koichi Wada [3] proposed a parallel PSO algorithm, called delayed exchange parallelization, which improves performance of PSO on distributed environment by hiding communication latency efficiently. By overlapping communication with computation, the proposed algorithm extracts parallelism inherent in PSO.

A field-programmable gate array (FPGA)-based parallel meta-heuristic particle swarm optimization algorithm (PPSO) and its application to global path planning for autonomous robot navigating in structured environments with obstacles is presented by Hsu-Chih Huang [4]. This PPSO consists of three parallel PSOs along with a communication operator in one FPGA chip. The parallel computing architecture takes advantages of maintaining better population diversity and inhibiting premature convergence in comparison with conventional PSOs.

Parsopoulos [5] proposed to delay the partial best fitness exchanging to N step later, which alleviates the dependency between each iteration and make it possible to overlap the communication overhead by calculation time.

The master–slave (MS) model is among the most popular approaches for parallel computing. In general, the MS model consists of one master process and M slave processes. The slave processes typically execute the following sequence of operations:

(1) **Receive** a particle and the buffer vector.

(2) **Compute** the objective value of the particle.

(3) **Send** the result to the master process.

The expected gain of an MS parallel model is the significant reduction of the total wall-clock time required by the algorithm.

A parallel master–slave model of the recently proposed cooperative micro-particle swarm optimization approach is introduced by Konstantinos E. Parsopoulos. The algorithm is based on the decomposition of the original search space in subspaces of smaller dimension. Each subspace is probed by a subswarm of small size that identifies suboptimal partial solution components. A context vector that serves as repository for the best attained partial solutions of all subswarms is used for the evaluation of the particles.

III. Description of Related Algorithms

This section presents some algorithms which are used in *experiments and results* section.

A. EPSO

The EPSO algorithm proposed by J.Barrera and Carlos A.C in 2009 [6]. In this method they have tried to solve optimization problems by modifying the mechanism of selecting the global optimum in PSO algorithm. By applying Coulomb rule shown in equation (1), a particle to be selected separately as the global optimum for each particle is calculated. In fact, it may be particles move toward different particles as global optimum, it means that for each particle it is possible that the global optimum be different and this leads to this fact that particles aggregate around local optimum instead of global optimum. It is obvious that more particles are aggregated around a point with better cost function value. The feature of this formula is that particles tend to move toward a point with suitable distance to that particle in addition to the suitability of its cost function.

$$F_{(j,i)} = \frac{1}{4\pi\varepsilon_0}.\frac{Q_1.Q_2}{r^2} \quad (1)$$

where F is the amount of electrostatic force and Q_1 and Q_2 are point load and finally $\frac{1}{4\pi\varepsilon_0}$ is called Coulomb constant.

By inspiring Coulomb rule, equation (2) is obtained to calculate the force between two particles in PSO.

$$F_{(j,i)} = \alpha.\frac{f(p_j).f(p_i)}{||p_j - p_i||^2} \quad (2)$$

where $f(p_j)$ is current particle's personal best value and $f(p_i)$ is the personal best value of a particle which has the possibility of moving toward that. The distance between particles i and j is computed in denominator. The value of α considered as Coulomb constant is calculated by equation (3) presented in article [7].

$$\alpha = \frac{||s||}{f(p_g) - f(p_w)} \quad (3)$$

where $||s||$ is scaling factor which is assumed as problem space, $f(p_g)$ equals to global optimum and $f(p_w)$ equals to the value of worst cost function in current population. As a result, particle p_j will move toward a particle from $\vec{\imath}$ with the highest value F. At each iteration, the position and velocity of a particle are updated by following two simple rules shown in equations:

$$v_t = w.v_{t-1} + R_1.c_1.\left(p_i^{best} - x_i\right) + R_2.c_2.(p^F - x_i) \quad (4)$$
$$x_i = x_i + v_i$$

B. FERPSO

FERPSO [7] is a commonly used algorithm presented to solve MMO problems. This algorithm has been proposed by Xiaodong Li in 2007. We could describe the behavior of this algorithm in nature point of view in this way that where there are more food, more birds are aggregated there. In fact, if birds find suitable food near them, they would not go toward many resources in far points. In this algorithm by applying equation (1) a particle to be selected as a global optimum for each separate particle is calculated. In fact, the general structure of FERPSO and EPSO are very similar to each other and both have the same complexity.

$$FER_{(j,i)} = \alpha.\frac{f(\vec{p}_j) - f(\vec{x}_i)}{|p_j - x_i|} \quad (5)$$

where $f(\vec{x}_i)$ is the value of particle's computing cost function in the current population and $f(p_j)$ is the particle's best personal value which has the possibility to move toward that. The dominator also computes the distance between personal best particle where there is possibility of motion toward that and the computing particle in the current population. α is also computed as equation (3). As a result, particle x_j will move toward particle of $\vec{\imath}$ which has the highest value of FER.

At each iteration, the position and velocity of a particle are updated by following two simple rules shown in equations:

$$v_t = w.v_{t-1} + R_1.c_1.\left(p_i^{best} - x_i\right) + R_2.c_2.(p^{FDR} - x_i) \quad (6)$$
$$x_i = x_i + v_i$$

IV. Proposed Method

In this section, basics of proposed method, named Parallel Reverse Niching Particle Swarm optimization (PRNPSO) will be presented.

The existing PSO-algorithm-based niching methods [8] employed the local version PSO_{lb} to solve the real-valued multimodal functions, where a subswarm or the neighborhood of a particle is identified by the similarity in the Euclidean metric. In contrast to finding the neighborhood which is highly time consuming, we propose to reversely niche the optimization function, to prevent converging to a single optimum. The proposed algorithm is also a parallel method which lets us to use power of parallel CPUs.

Fig-1 shows flowchart of proposed algorithm. As it can be taken from the figure, the proposed method consists of many PSO-populations which work in parallel. Each PSO has its own population and continue its own way to evolution. But they exchange their global bests in every iteration. Global best is used to calculate penalty of particles which is the novelty of this work. Penalty function applies reverse niching which turns unimodal PSO algorithm to a multimodal optimization algorithm.

As we mentioned earlier, this paper attempts to solve multimodal optimization problem by reverse niching. Reverse niching is done by penalty function. Penalty function uses other populations' global bests as center of Gaussian functions. Penalty of a point X for population j can be calculated as following:

$$P_j(X) = \sum_{i=1, i\neq j}^{p} f(GB_i)\, e^{-\frac{(X-GB_i)^2}{2c_j^2}} \qquad (7)$$

where GB_i is global best of i-th population and c_j is a constant which controls width of reverse niche. This penalty prevents particles of this population from approaching to global best of other populations.

As fig.1 shows, each population of the proposed algorithm works independently. They only exchange their global bests in each iteration. They totally work in parallel which let the algorithm to improve performance.

V. Experiments and Results

This section presents experiments to evaluate the proposed method, and their Results.

A. Test Function

In order to test the proposed niching algorithm, 5 commonly used multimodal bench mark functions are used. This functions and their characteristics are presented in Table I. As most of optimization algorithms have good performance on one-dimensional functions, two-dimensional functions, which are more challenging, have been considered and one-dimensional functions have been omitted.

B. Experimental Environment

All the algorithms are implemented using Matlab 2013 and executed on two computers with Intel core i5 and i7 CPU.

C. Experimental Results

Table II compares proposed PRNPSO method with FERPSO and EPSO methods. FERPSO and EPSO have been described in section III. Parameter settings for each test function in addition with average number of optima found in each method have been presented in Table II. Number of optima found have been averaged in 10 runs of each algorithm. For the PRNPSO method, number of populations and number of particles for each population have been set in a manner that total number of particles in swarm is equal to the number of particles in other methods, i.e. "number of populations" times "number of particles in each population" is equal to the "number of particles" presented in Table II. More specific, we set the parameter "number of population" to the half of the "number of particles" in column 2 of the Table II and the parameter "number of particles in each population" to "2", i.e. there are two particles in each "*Particle*/2" population of swarm. This setting lets the algorithm to find optima as much as possible, because each population will lead the algorithm to one optima. Therefore, to increase number of optima found by PRNPSO, number of populations should be increased.

Table II shows that PRNPSO outperforms FERPSO in all cases. But EPSO is not dominated by PRNPSO in all cases. PRNPSO is better than EPSO when particle size is small. But by increasing number of particles EPSO reaches to better performance and outperforms PRNPSO.

VI. Conclusion

In this paper a new method named Parallel Reverse Niching PSO (PRNPSO) is presented for multimodal optimization. PRNPSO is compared with two other algorithms: FERPSO and EPSO. All algorithms are tested on 5 widely used two-dimensional test functions with different number of particles and iterations. Results show that EPSO and PRNPSO are better than FERPSO based on number of optima found. PRNPSO is better than EPSO when number of particles is small. By increasing number of particles, EPSO outperforms PRNPSO.

Width of Reverse Niching (parameter c_j in formula (7)) is an input parameter for the algorithm which can effect performance of the algorithm. Finding best value for c_j needs more experiments which is one of our future works. Another future work is to find value for c_j or at least to find a way to adaptively find a good value for it.

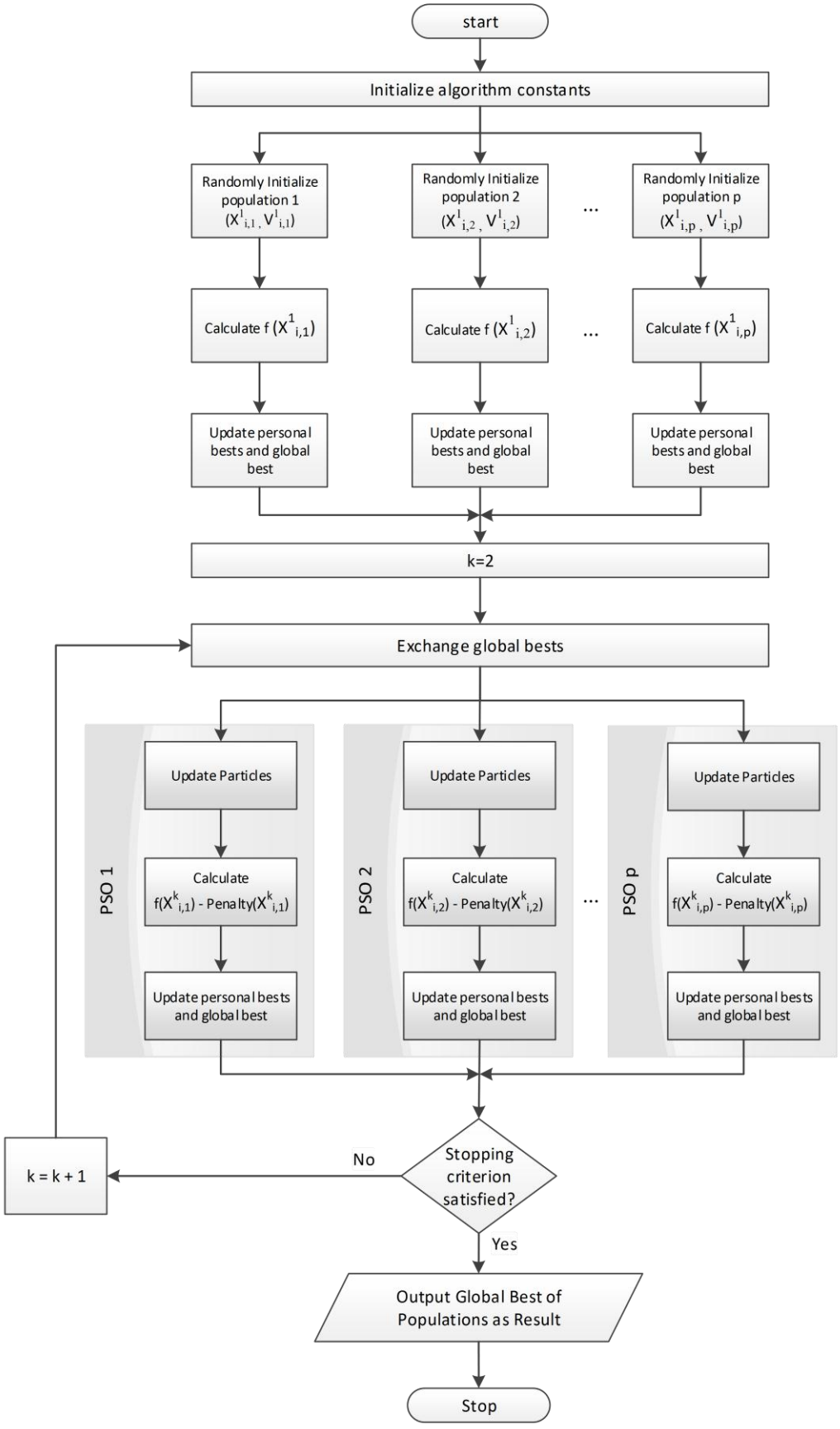

Fig. 1 Flowchart of Parallel Reverse Niching PSO

TABLE I
TEST FUNCTIONS HAVE BEEN USED FOR EXPERIMENTS.

Name	Test Function	Range
$f1$: Six-hump camel back	$f(x_1,x_2)=\left(4-2.1x_1^2+\frac{x_1^4}{3}\right)+x_1^2+x_1x_2+(-4+4x_2^2)x_2^2$	$-1.9<x_1<1.9$ $-1.1<x_2<1.1$
$f2$: Ackley	$f(x_1,x_2)=-20.\exp(-2.\sqrt{\frac{1}{2}({x_1}^2+{x_2}^2)}-\exp(\frac{1}{2}(\cos(cx_1)+\cos(cx_2))+20+\exp(1))$	$-5<x_1<5$ $-5<x_2<5$
$f3$: Rastrigin	$f(x_1,x_2)=10n+\sum_{i=1}^{n}10.2+[x_i^2-10\cos(2\pi x)]+[y^2-10\cos(2\pi y)]$	$-5.12<x_1<5.12$ $-5.12<x_2<5.12$
$f4$: Shubert	$f(x_1,x_2)=-\sum_{i=1}^{5}i\cos(i+1)\,x_1+1].\sum_{i=1}^{5}i\cos(i+1)\,x_2+1].$	$-5.12<x_1<5.12$ $-5.12<x_2<5.12$
$f5$: Fifth function of DeJong	$f(x_1,x_2)=\{0.002+\sum_{j=1}^{25}\left[j+(x_1-a_{1j})^6+(x_2-a_{2j})^6\right]^{-1}\}^{-1}$ $(a_{ij})=\begin{pmatrix}-32 & -16 & 0 & 16 & 32 & -32 & 0 & 16 & 32\\ -32 & -32 & -32 & -32 & -32 & -16 & \cdots & 32 & 32 & 32\end{pmatrix}$	$-40<x_1<40$ $-40<x_2<40$

TABLE II
AVERAGE NUMBER OF OPTIMA FOUND IN 10 RUN OF ALGORITHM FOR EACH TEST FUNCTION.

Function	Number of Particles	Number of Iterations	FERPSO	EPSO	PRNPSO
$f1$	30	60	2.10	2.90	2.6
	60	60	2.30	4.30	3.3
$f2$	200	70	11.40	26.20	28.9
	400	70	11.50	46.30	36.9
	1000	50	17.50	81.20	49.5
$f3$	200	70	10.09	27.18	40.9
	400	80	17.63	54.10	53.9
	1000	60	22.72	88.00	69.8
$f4$	200	70	-	26.50	33.5
	400	80	-	55.80	53.7
	1000	55	-	91.40	80.5
$f5$	200	30	9.40	19.40	20.8
	400	30	17.00	24.10	24.3

REFERENCES

[1] R. E. J. Kennedy, "Particle swarm optimization," (1995) 1942–1948.

[2] J. F. R. J.-S. P. Shu-Chuan Chu, "A Parallel Particle Swarm Optimization Algorithm with Communication Strategies".

[3] K. W. Bo Li, "Communication latency tolerant parallel algorithm for particle swarm optimization," Elsevier- Parallel Computing, p. 1–10, 2011.

[4] H.-C. Huang, "FPGA-Based Parallel Metaheuristic PSO Algorithm and Its Application to Global Path Planning," Springer, 2013.

[5] K. E. Parsopoulos, "Parallel cooperative micro-particle swarm optimization: A master–slave model," Elsevier, 2012.

[6] Julio Barrera and Carlos A. Coello Coello, "A Particle Swarm Optimization Method for Multimodal Optimization Based on Electrostatic Interaction," *Lecture Notes in Computer Science-MICAI 2009: Advances in Artificial Intelligence*, vol. 5845, pp. 622-632, 2009.

[7] Xiaodong Li, "A multimodal particle swarm optimizer based on fitness Euclidean-distance ratio," in *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation*, London, 2007.

[8] B. Qu, J.J. Liang and P.N. Suganthan, "Niching particle swarm optimization with local search for multi-modal optimization," vol. 197, pp. 131-143, 2012.