International Journal of Electronics Signals and Systems

Volume 3 | Issue 2 Article 5

October 2013

Implementation of a Simplified Cultural-Based Multi-Objective Particle Swarm Optimization

Anupam Francis
Electronics and Communication Department Amrita School of Engineering, Coimbatore, India,
anupamfrancis@gmail.com

N. Mohan Kumar
Electronics and Communication Department Amrita School of Engineering, Coimbatore, India,
mk.mohankumar@gmail.com

M. Nirmala Devi
Electronics and Communication Department Amrita School of Engineering, Coimbatore, India,
nirmala_amrita@yahoo.co.in

Implementation of a Simplified Cultural-Based Multi-Objective Particle Swarm Optimization

Anupam Francis[1], Y.Kirthika, Sidharth Mohan,
Mohammed Nijil, Elakiya A, Sreedivya P
Electronics and Communication Department
Amrita School of Engineering,
Coimbatore, India
Email : [1]anupamfrancis@gmail.com

Mr. N Mohan Kumar[2], Dr. M Nirmala Devi[3]
[2]Assistant Professor, [3]Associate Professor
Electronics and Communication Department
Amrita School of Engineering,
Coimbatore, India
Email: [2]mk.mohankumar@gmail.com
[3]nirmala_amrita@yahoo.co.in

Abstract— **This paper presents a simplified Cultural based Multi-Objective Particle Swarm Optimization (MOPSO) algorithm. In this algorithm we modify momentum and global acceleration components of the conventional MOPSO algorithm. The algorithm has been tested on common benchmark functions. Its performance has been compared with other algorithms, using standard test metrics. The results show that the cultural based MOPSO is more efficient and robust.**

Keywords- Particle Swarm Optimization, Multi-Objective Particle Swarm Optimization, Cultural Algorithm, Kursawe, ZDT1

I. INTRODUCTION

Particle Swarm Optimization [1] (PSO) is a population based stochastic optimization technique that can solve problems with one objective. Multi Objective Particle Swarm Optimization [2] (MOPSO) extends the capability of PSO algorithm by being able to solve multi-objective problems. A Genetic Algorithm [3] (GA) is a robust problem-solving method based on natural selection. MOPSO can be further modified by adding constraints to the algorithm and this will in turn give us faster and more accurate results.

We analyzed several MOPSO papers. These are Speed-constrained multi-objective particle swarm optimization [4] (SMPSO), Dynamic multi-objective particle swarm optimization [5] (DMOPSO), Dynamic swarm multi-objective particle swarm optimization [6] (DSMOPSO) and Cultural-based multi-objective particle swarm optimization [7]. A comparative study of cultural MOPSO was done with other algorithms and we infer that cultural MOPSO is better in both terms of efficiency and computational cost. We propose to implement the topographical knowledge, as described in the cultural MOPSO paper, in our algorithm, in order to improve performance.

In section II, we briefly describe some of the MOPSO algorithms that we have studied. Section III details the algorithm of the simplified cultural-based MOPSO algorithm. Section IV describes the various test metrics and test functions that have been used. Section V contains the results that we have obtained.

II. BACKGROUND

A. PSO

Particle swarm optimization (PSO) is a population based stochastic optimization technique. It was developed by Dr.Kennedy and Dr.Eberhart in 1995 as a stylized representation of the movement of organisms in bird flock or a fish school. In past several years, PSO has been successfully applied in many research and application areas. It has been demonstrated that PSO gets better results in a faster, cheaper way compared with other methods. Another reason that PSO is attractive is that there are few parameters to adjust. One version, with slight variations, works well in a wide variety of applications. Particle swarm optimization has been used for approaches that can be used across a wide range of applications, as well as for specific applications focused on a specific requirement.

The drawbacks of PSO are that the performance may not be competitive in some problems and that the representation of the weights is difficult and the genetic operators have to be carefully selected or developed.

B. MOPSO

Many real world applications have more than one objective that requires optimization. It is hard to model these objectives as a single function and these problems are hence modeled as multi-objective functions. The task of finding one or more optimal solutions is known as multi-objective optimization. The selection of the *pbest* and *gbest* criteria is the greatest challenge in extending the PSO to multi-objective as there is a set of optimal solutions rather than a unique optimum in multi objective problems.

Personal best position *(pbest)* defines the best position found by the particle. It is updated whenever the particle reaches a position with a better fitness value than the fitness value of the previous personal best. MOPSO finds the best possible solutions which satisfy all the objectives and constraints and takes *Pareto dominance* into account while searching for

solutions. There isn't a single best solution but a set of optimal solutions which is called as the pareto-optimal set.MOPSO does not have the drawbacks of PSO. It allows us to get a set of optimum solutions for a problem having multiple variables.

C. Algorithm for conventional MOPSO

The algorithm of MOPSO is the following:

1. Initialize the population POP:
   (a) FOR i=0 TO MAX (MAX = number of particles)
   (b)INITIALIZE POS()
2. Initialize the velocity of each particle:
   (a) FOR i=0 TO MAX
   VEL(i) = 0
3. Evaluate each of the particles in POS.
4. Store non-dominated vectors in the repository REP.
5. Generate hypercubes of the search space explored so far. Using these hyper-cubes as a coordinate system, locate the particles. Each particle's co-ordinates are defined according to the values of its objective functions.
6. Initialize the memory of each particle (serves as a guide to travel through the search space. Also stored in the repository):
   (a) FOR i=0 TO MAX
   (b) PBEST(i) = POS(i)
7. WHILE maximum number of cycles has not been reached DO:

(a) Compute the speed of each particle using the following expression:

$$VEL(i) = w \times VEL(i) + C_1\left(PBEST(i) - POS(i)\right) + C_2\left(REP(h) - POS(i)\right) \quad (2.1)$$

where w (inertia weight) takes a value of 0.4; C_1 and C_2 are random numbers in the range [0..1]; *PBESTS(i)* is the best position that the particle *i* has had 2 ; *REP(h)* is a value that is taken from the repository; the index h is selected in the following way: those hypercubes containing more than one particle are assigned a fitness equal to the result of dividing any number x > 1 (we used x = 10 in our experiments) by the number of particles that they contain. This aims to decrease the fitness of those hypercubes that contain more particles and it can be seen as a form of fitness sharing. Then roulette-wheel selection using these fitness values is applied in order to select the hypercube from which we will take the corresponding particle. Once the hyper-cube has been selected, a particle within such a hypercube is randomly selected. POP(i) is the current value of the particle *i*.

(b) Compute the new positions of the particles adding the speed produced from the previous step:

$$POS(i) = POS(i) + VEL(i) \quad (2.2)$$

(c) Maintain the particles within the search space in case they go beyond its boundaries (avoid generating solutions that do not lie on valid search space).

(d) Evaluate each of the particles in POS.

(e) Update the contents of REP together with the geographical representation of the particles within the hypercubes. This update consists of inserting all the currently non-dominated locations into the repository. Any dominated locations from the repository are eliminated in the process. Since the size of the repository is limited, whenever it gets full, we apply a secondary criterion for retention: those particles located in less populated areas of objective space are given priority over those lying in highly populated regions.

(f) When the current position of the particle is better than the position contained in its memory, the particle's position is updated using:

$$PBESTS(i) = POS(i) \quad (2.3)$$

The criterion to decide what position from memory should be retained is simply to apply Pareto dominance (i.e., if the current position is dominated by the position in memory, then the position in memory is kept; otherwise, the current position replaces the one in memory; if neither of them is dominated by the other, then we select one of them randomly).

(g) Increment the loop counter

8. $END\ WHILE$

D. Speed-constrained Multi-objective particle swarm optimization[4]

A multi-objective particle swarm optimization algorithm characterized by the use of a strategy to limit the velocity of the particles is presented. The proposed approach is called Speed-constrained Multi-objective PSO (SMPSO). It allows producing new effective particle positions in those cases in which the velocity becomes too high. Other features include the use of polynomial mutation as a turbulence factor as well as an external archive to store the non-dominated solutions found during the search. The proposed approach is compared with five other multi-objective algorithms which are state-of-the-art in the area. Two different criteria are adopted for the comparison: the quality of the resulting approximation sets, and the convergence speed to the Pareto front. The experiments carried out indicate that SMPSO obtains better results in terms of both, accuracy and speed.

E. Cultural MOPSO [7]

Most MOPSOs use fixed momentum and acceleration for all particles throughout the evolutionary process. In this paper, a cultural framework to adapt the personalized flight parameters of the mutated particles in MOSPSO is introduced. There is a need for a personalized weight for each particle. Cultural algorithm is a computational frame work which consists of two different spaces - population space and belief space. Belief space is the information which does not depend on the individuals who can be generated and accessed by the members of population space. It consists of different types of information called knowledge. The different knowledges are

1. Situational knowledge - a set of exemplary individuals useful for the experiences of all the individuals

2. Normative knowledge - consists of a set of promising range.
3. Topographical knowledge - keeps track of the best individuals which have been found so far in the promising region.

In cultural MOPSO, first the population space and corresponding belief space is initialized. Then we evaluate the population space using fitness values, and hence the belief space. After the belief space is updated, the corresponding knowledge should be used to influence the MOPSO parameters. They are:

1. Adapting global acceleration: the topographical knowledge is used to adapt the local acceleration. It adjusts the direction and step size in the global acceleration
2. Adapting local acceleration: the situational knowledge is used to build local grids in order to adjust local acceleration.
3. Adapting momentum : the normative knowledge is used to adapt the momentum of particles
4. Gbest selection: the topographical knowledge is used stored in belief space to select gbest in each iteration.
5. Pbest selection: the situational knowledge is used to select pbest.

A comparative study of cultural MOPSO is done with other algorithms and we infer that cultural MOPSO is better in terms of efficiency.

III. SIMPLIFIED CULTURAL MOPSO

We have developed a simplified version of the cultural MOPSO algorithm that adapts momentum and global acceleration. We decided that the process concerning the adapting of local acceleration introduces a new level of complexity as it involves variable memory sizes for each particle.

Our proposed algorithm is as follows:

1. Initialize the population POP:
   (a) $FOR\ i\ =\ 0\ TO\ MAX$
   (MAX = number of particles)
   (b) $Initialize\ POP()$
2. Initialize the velocity of each particle:
   (a) $FOR\ i\ =\ 0\ TO\ MAX$
   (b) $VEL(i)\ =\ 0$
3. Evaluate each of the particles in POP.
4. Store non-dominated vectors in the repository REP.
5. Generate hypercubes of the search space explored so far. Using these hyper-cubes as a coordinate system, locate the particles. Each particle's co-ordinates are defined according to the values of its objective functions.
6. Initialize the memory of each particle (serves as a guide to travel through the search space. Also stored in the repository):
   (a) $FOR\ i\ =\ 0\ TO\ MAX$
   (b) $PBESTS(i)\ =\ POP(i)$
7. Initialize the momentum w along each dimension or variable of the searchspace. Our initial w was taken as 0.4 for each dimension.
8. Initialize the global acceleration coefficient c_g. We took its initial value as 2.
9. WHILE maximum number of cycles has not been reached DO:
(a) Compute the speed of each particle using the following expression

$$VEL(i) = w.VEL(i) + C_p.R_1\big(PBEST(i) - POP(i)\big) + C_g.R_2\big(REP(h) - POP(i)\big) \quad (3.1)$$

where *w* is the inertia weight; R_1 and R_2 are random numbers in the range [0..1]; *PBESTS(i)* is the best position that the particle *i* has had; *REP(h)* is a value that is taken from the repository; the index *h* is selected in the following way: those hypercubes containing more than one particle are assigned a fitness equal to the result of dividing any number x (we used x = 1 in our experiments) by the number of particles that they contain. This aims to decrease the fitness of those hypercubes that contain more particles and it can be seen as a form of fitness sharing. Then roulette-wheel selection using these fitness values is applied in order to select the hypercube from which we will take the corresponding particle. Once the hyper-cube has been selected, a particle within such a hypercube is randomly selected. POP(i) is the current value of the particle *i*.
(b) Compute the new positions of the particles adding the speed produced from the previous step:

$$POP(i)\ =\ POP(i)\ +\ VEL(i) \quad (3.2)$$

(c) Maintain the particles within the search space in case they go beyond its boundaries (avoid generating solutions that do not lie on valid search space).
(d) Evaluate each of the particles in POP.
(e) Update the contents of REP together with the geographical representation of the particles within the hypercubes. This update consists of inserting all the currently non-dominated locations into the repository. Any dominated locations from the repository are eliminated in the process. Since the size of the repository is limited, whenever it gets full, we apply a secondary criterion for retention: those particles located in less populated areas of objective space are given priority over those lying in highly populated regions.
(f) When the current position of the particle is better than the position contained in its memory, the particle's position is updated using:

$$PBESTS(i)\ =\ POP(i) \quad (3.3)$$

The criterion to decide what position from memory should be retained is simply to apply Pareto dominance (i.e., if the current position is dominated by the position in memory, then the position in memory is kept; otherwise, the current position replaces the one in memory; if neither of them is dominated by the other, then we select one of them randomly).

(g) Determine if the global leader REP(h) is moving out of a crowded area or into a crowded area. In order to promote diversity, we want to support the movement out of a crowded area and hinder movement into a crowded area. Hence, we modify the global acceleration coefficient as follows.

$$C_g(i+1)=\begin{cases} C_p(i) + \alpha(|N(i+1)-N(i)|), \text{ if } N(i+1)>N(i) \\ C_g(i) - \alpha(|N(i+1)-N(i)|), \text{ if } N(i+1)<N(i) \end{cases} \quad (3.4)$$

Where α is a predetermined scaling factor and N(i) and N(i+1) refer to the populations of the hypercubes to which the global leader belongs to.

(h) Determine the maximum and minimum velocities of all particles stored in the repository. We wish to bring the velocities of all particles searching the searchspace to this preferable range.

Using this, we adapt the momentum or inertia of the particles:

$$W(i+1)=\begin{cases} W(i) - \Delta w, & \text{if } v(i)>v_{max} \\ W(i) \quad , & \text{if } v(i)>v_{min} \text{ and } v(i)<v_{max} \\ W(i)+ \Delta w, & \text{if } v(i)<v_{min} \end{cases} \quad (3.5)$$

Where Δw is a predetermined constant.

(i) Increment the loop counter

8. *END WHILE*

IV. Test Metrics and Test Functions

A. Test Metrics

There are various metrics that can be used to determine the performance of an algorithm. The metrics we have chosen are:

i. *Generational Distance (GD):*

Generational Distance is a way of estimating how far the elements in the set of non-dominated vectors found so far are from those in the Pareto optimal set (true Pareto front). Generational distance is defined as

$$GD = \frac{\sqrt{\sum_{i=1}^{n} d_i^2}}{n} \quad (4.1)$$

Where n is the number of vectors in the set of non-dominated solutions we have found and d_i is the Euclidean distance between each of these vectors and the nearest vector in the Pareto optimal set. The smaller the *GD*, the closer we are to the true Pareto front.

ii. *Spacing (S):*

A metric that determines how well the spread of non-dominated vectors is distributed is Spacing. Spacing is the distance variance of neighbouring vectors in the non-dominated set found so far. It is defined as

$$S \triangleq \sqrt{\frac{\sum_{i=1}^{n}(\bar{d}-d_i)^2}{n-1}} \quad (4.2)$$

Where n is the number of non-dominated vectors found so far, d_i is defined as

$$d_i = min_j\left(\left|f_1^i(\vec{x}) - f_1^j(\vec{x})\right| + \left|f_2^i(\vec{x}) - f_2^j(\vec{x})\right|\right) \quad (4.3)$$

And d is the mean of all d_i. A value of $S = 0$ means that all members of the Pareto front currently available are equidistantly spaced from each other.

B. Test functions

Test functions are multi-objective problems that have been used in a number of papers and which have clearly defined pareto fronts. The test functions we have chosen for our simulation are:

i. *Kursawe function:*

$$f1(\vec{x}) = \sum_{i=1}^{n-1} -10 \exp\left(-0{,}2\sqrt{x_i^2 + x_{i+1}^2}\right) \quad (4.4)$$

$$f2(\vec{x}) = \sum_{i=1}^{n-1}(|x_i|^{0{,}8} + 5 \sin(x_i)^2) \quad (4.5)$$
$$-5 \le x1, x2, x3 \le 5$$

ii. *Deb bimodal function*:

$$f2(x1, x2) = \frac{g(x2)}{x1} \quad (4.6)$$

$$f1(x1, x2) = x1 \quad (4.7)$$
$$0{,}1 \le x1 \le 1{,}0 \ \& \ 0{,}1 \le x2 \le 1{,}0$$
$$g(x2) = 2{,}0 - \exp\left\{-\left(\frac{x_2 - 0{,}2}{0{,}004}\right)^2\right\}$$
$$- 0{,}8 \exp\left\{-\left(\frac{x_2 - 0{,}6}{0{,}004}\right)^2\right\} \quad (4.8)$$

iii. *ZDT1 function:*

The ZDT1 function is the first in a family of 6 ZDT functions that are standard test functions. ZDT1 is defined as

$$g(x_2, x_3, \dots, x_n) = 1 + 9\frac{(\sum_{i=2}^{n} x_i)}{(n-1)} \quad (4.9)$$

$$h(f_1, g) = 1 - \sqrt{\frac{f1}{g}} \quad (4.10)$$

$$f1(x) = x_1 \quad (4.11)$$

$$f2(x) = g(x_2, x_3, \dots, x_n)h(f_1, g) \quad (4.12)$$

True pareto fronts of kursawe and Deb bimodal function are shown in figure 4.1 and 4.2 .True pareto front of ZDT1 function is shown in figure 4.3.

V. RESULTS AND DISCUSSION

A. Test function 1 – Kursawe:

We used the Kursawe function as the first test function. We performed our experiment using the conventional MOPSO, SMPSO and our simplified cultural MOPSO algorithms. We recorded the time taken to complete an iteration as well as the generational distance and spacing.

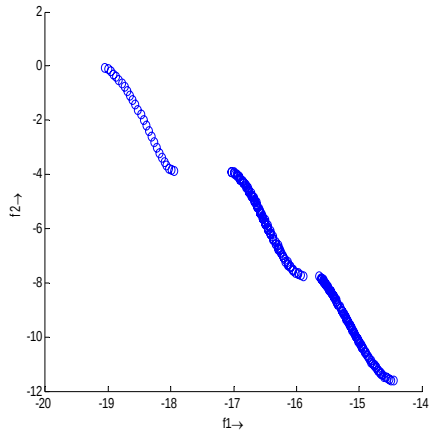

Figure 4.1. True pareto front of Kursawe function

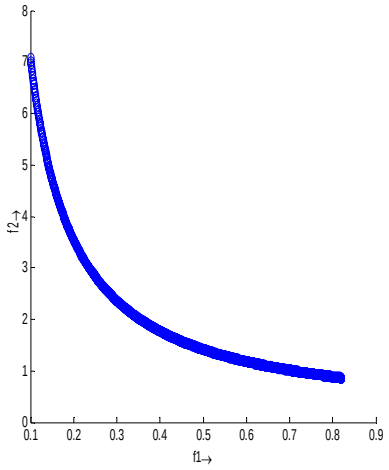

Figure 4.2. True pareto front of Deb Bimodal function

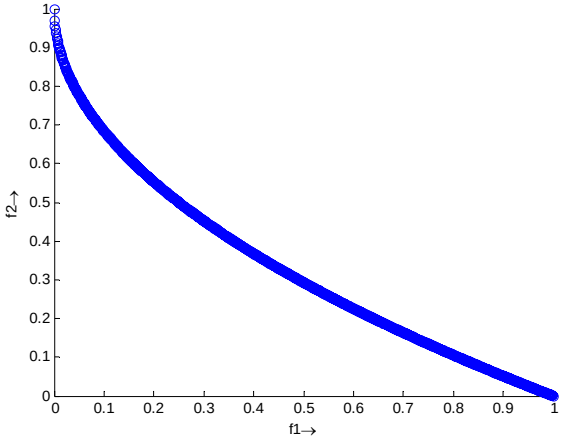

Figure 4.3. True pareto front of ZDT1 function

TABLE I
COMPUTATIONAL TIME (IN SECONDS) REQUIRED BY EACH ALGORITHM FOR THE FIRST TEST FUNCTION

TIME	MOPSO	SMPSO	S-CULTURAL
BEST	0.2662455	0.2564965	0.158833
WORST	0.3242745	0.2880685	0.2589375
AVERAGE	0.2824305	0.2734015	**0.2430815**
MEDIAN	0.28067	0.274094	0.2557355
STANDARD DEVIATION	0.0172415	0.0113775	0.0304765

TABLE II
RESULTS OF THE GENERATIONAL DISTANCE METRIC FOR THE FIRST TEST FUNCTION

GD	MOPSO	SMPSO	S-CULTURAL
BEST	0.00016319	0.00027643	0.00035147
WORST	0.022876	0.00095289	0.0014282
AVERAGE	0.0025665	**0.00050854**	0.00065142
MEDIAN	0.00031323	0.00040894	0.00041919
STANDARD DEVIATION	0.007137	0.00023394	0.0003954

TABLE III
RESULTS OF THE SPACING METRIC FOR THE FIRST TEST FUNCTION

SPACING	MOPSO	SMPSO	S-CULTURAL
BEST	0.022055	0.030725	0.033412
WORST	0.2822	0.0618	0.068069
AVERAGE	0.054904	**0.040871**	0.045387
MEDIAN	0.031433	0.036078	0.038985
STANDARD DEVIATION	0.080054	0.010593	0.011981

The results are plotted in Tables I, II and III. From Table I, we see that the simplified cultural algorithm runs faster. But with respect to GD and spacing, SMPSO performs slightly better, which we observe from Tables II and III.

B. Test function 2: ZDT1:

We then used the ZDT1 function to test the performance of the simplified cultural MOPSO algorithm. We again recorded the time taken, generational distance and spacing. The algorithm was compared with conventional MOPSO and SMPSO as before, and the results are in Tables IV, V and VI. From Table IV, we see that S-cultural performed much faster than conventional MOPSO and SMPSO. It is almost 4 times faster than conventional MOPSO and more than 2 times faster than SMPSO. From Table V, we see that the GD of the results obtained from S-cultural is better than those of conventional MOPSO and SMPSO. From Table VI, we see that spacing too is better for S-cultural than those of the other two algorithms.

TABLE IV
COMPUTATIONAL TIME (IN SECONDS) REQUIRED BY EACH ALGORITHM FOR THE SECOND TEST FUNCTION

TIME	MOPSO	SMPSO	S-CULTURAL
BEST	0.5034805	0.231891	0.0629795
WORST	0.616785	0.361459	0.5602935
AVERAGE	0.540337	0.292002	**0.14227**
MEDIAN	0.53579	0.28668	0.074129

STANDARD DEVIATION	0.037909	0.0369305	0.1586325

TABLE V
RESULTS OF THE GENERATIONAL DISTANCE METRIC FOR THE SECOND TEST FUNCTION

GD	**MOPSO**	**SMPSO**	**S-CULTURAL**
BEST	0.094968	0.096185	3.5153e-006
WORST	0.3149	0.42546	0.31823
AVERAGE	0.19076	0.22281	**0.10519**
MEDIAN	0.1845	0.18486	0.081759
STANDARD DEVIATION	0.066606	0.1174	0.10963

TABLE VI
RESULTS OF THE SPACING METRIC FOR THE SECOND TEST FUNCTION

SPACING	**MOPSO**	**SMPSO**	**S-CULTURAL**
BEST	0.15234	0.14366	0.0036621
WORST	0.22571	0.37547	0.21235
AVERAGE	0.18644	0.241	**0.11356**
MEDIAN	0.18579	0.21022	0.11732
STANDARD DEVIATION	0.019952	0.08654	0.077437

C. Test function 3 – Deb bimodal

The third test function we used was the Deb bimodal function. The results obtained from running the conventional MOPSO, SMPSO and S-cultural are shown in Tables VII, VIII and IX. From Table VII, we see that S-cultural is faster than the other two algorithms. From Tables VIII and IX, we see the S-cultural outperforms conventional MOPSO and SMPSO by a huge margin. The GD is more than ten times better than either of the other two algorithms. The spacing is more than 6 times better than either of the other two algorithms.

.

TABLE VII
COMPUTATIONAL TIME (IN SECONDS) REQUIRED BY EACH ALGORITHM FOR THE THIRD TEST FUNCTION

TIME	**MOPSO**	**SMPSO**	**S-CULTURAL**
BEST	0.520726	0.292909	0.30119
WORST	0.7403705	0.749493	0.509036
AVERAGE	0.6798645	0.594431	**0.4372145**
MEDIAN	0.7261335	0.653938	0.4457075
STANDARD DEVIATION	0.080689	0.1677665	0.0579735

TABLE VIII
RESULTS OF THE GENERATIONAL DISTANCE METRIC FOR THE THIRD TEST FUNCTION

GD	**MOPSO**	**SMPSO**	**S-CULTURAL**
BEST	0.00069252	1.0388e-005	0.00026567
WORST	0.59595	0.62107	0.34644
AVERAGE	0.37233	0.41091	**0.035391**
MEDIAN	0.48226	0.47846	0.00094371
STANDARD DEVIATION	0.26032	0.22983	0.10929

TABLE IX
RESULTS OF THE SPACING METRIC FOR THE THIRD TEST FUNCTION

SPACING	**MOPSO**	**SMPSO**	**S-CULTURAL**
BEST	0.049129	0.0047963	0.029069
WORST	1.2645	1.3697	0.81271
AVERAGE	0.82582	0.9201	**0.13333**
MEDIAN	1.0867	1.0556	0.06232
STANDARD DEVIATION	0.52963	0.47043	0.23908

VI. CONCLUSION

On the basis of our experiments and results, we see that the simplified cultural MOPSO algorithm performs better the conventional MOPSO and Speed constrained MOPSO (SMPSO) algorithms. This improvement in performance is attributed to the personalization of momentum and acceleration coefficients. Further work can be in determining the combination of parameters that provide the best results across a wide variety of test functions.

REFERENCES

[1] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in Proc. Int. Joint Conf. Neural Netw., Perth, Australia, 1995, pp. 1942–1948.
[2] C.A. Coello Coello and M.S. Lechuga,"MOPSO: A proposal for multiple objective particle swarm optimization", in Congress on Evolutionary Computation, pages 825-830. IEEE, 2002.
[3] S. L. Ho, S. Yang, G. Ni, E. W. C. Lo, and H. C. Wong, "A particle swarm optimization based method for multiobjective design optimizations," IEEE Trans. Magn., vol. 41, no. 5, pp. 1756–1759, May 2005.
[4] Nebro, A.J.; Durillo, J.J.; Garcia-Nieto, J.; Coello Coello, C.A.; Luna, F.; Alba, E.; SMPSO: A New PSO-based Metaheuristic for Multi-objective Optimization.; *Univ. of Malaga, Malaga, Computational intelligence in multi-criteria decision-making, IEEE symposium on computation intelligence,2009.*
[5] W. F. Leong and G. G. Yen, "PSO-based multiobjective optimization with dynamic population size and adaptive local archives," *IEEE Trans. Syst.,Man, Cybern. B, Cybern., vol. 38, no. 5, pp. 1270–1293, Oct. 2008.*
[6] G. G. Yen and W. F. Leong, "Dynamic multiple swarms in multiobjective particle swarm optimization," *IEEE Trans. Syst., Man, Cybern. A, Syst.,Humans, vol. 39, no. 4, pp. 890–911, Jul. 2009.*
[7] Daneshyari, M.; Yen, G.G.; Dept. of Electr. & Comput. Engg, Oklahoma State Univ., Stillwater, OK, USA : Cultural-Based Multi-objective Particle Swarm Optimization; ," IEEE Transactions on System,Man, Cybern. B, Cybern., vol. 41, sept. 2010.
[8] Carlos A. Coello Coello, *Member, IEEE*, Gregorio Toscano Pulido, and Maximino Salazar Lechuga, "Handling Multiple Objectives With Particle Swarm Optimization", *IEEE transactions on evolutionary computation, vol. 8, no. 3, June 2004*