International Journal of
Intelligent Systems and
Applications in Engineering
ISSN:2147-6799 www.atscience.org/IJISAE
Advanced Technology and Science
Original Research Paper

An Analysis of Archive Update for Vector Evaluated Particle Swarm Optimization

Zuwairie Ibrahim*[1], Lim Kian Sheng, Faradila Naim, Mohd Falfazli Mat Jusof , Nurul Wahidah Arshad

Accepted 15h August 2014 **DOI: 10.18201/ijisae.48588**

Abstract: Multi-objective optimization problem is commonly found in many real world problems. In computational intelligence, Particle Swarm Optimization (PSO) algorithm is a popular method in solving optimization problems. An extended PSO algorithm called Vector Evaluated Particle Swarm Optimization (VEPSO) has been introduced to solve multi-objective optimization problems. VEPSO algorithm requires an archive, which is used to record the solutions found. However, the outcome may be differ depending on how the archive is used. Hence, in this study, the performance of VEPSO algorithm when updates the archive at different instances is investigated by measuring the convergence and diversity by using standard test functions. The results show that the VEPSO algorithm performs better when update the archive during the search process, in the iterations.

Keywords: *Multi-objective, Optimization, Particle Swarm Optimization, Vector-Evaluated, Archive*

1. Introduction

Particle Swarm Optimization (PSO) algorithm, which has been proposed by James Kennedy and Russell Eberhart [1], has getting more attentions due to its simplicity in solving optimization problems [2-3]. PSO algorithm is inspired by the social behavior of bird flocking and fish schooling to find the optimum solution.
Since the original PSO algorithm is basically introduced to solve single-objective optimization (SOO) problems, for solving multi-objective optimization (MOO) problems, a number of extended PSO algorithms, such as Dynamic Neighborhood PSO [4], Multi-Objective PSO (MOPSO) [5], Another MOPSO (AMOPSO) [6], and Vector Evaluated Particle Swarm Optimization (VEPSO) [7] have been introduced. The VEPSO algorithm, which is motivated by Vector Evaluated Genetic Algorithm (VEGA) [8], requires multiple swarms in which each swarm optimizes one objective and the information regarding the best solution found in one swarm is transferred to the neighboring swarm. Besides, an archive is used to record the non-dominated solutions.
To date, the VEPSO algorithm has been successfully applied in various MOO problems such as supersonic ejector [9], antenna design [10], composite structure [11], DNA sequence design [12], and machine scheduling [13]. Even though the usefulness of VEPSO algorithm in solving these problems has been shown by many researchers, there is lack of quantitative performance evaluation carried out for understanding the performance of this algorithm and the effect of archive update at difference instance.
Hence, the main objective of this paper is to provide a quantitative performance measure for the VEPSO algorithm specifically in archive update at different instance. In this work, the performance measures used are the total number of non-dominated solutions found, *Generational Distance* [14], *Spread* [15], and Hypervolume [16]. Besides, the algorithm is tested on various standard benchmark test functions, which are ZDT [17], DTLZ [18], and WFG [19]. It is also worth to note that this paper considers continuous or real valued solution which has a continuous search space.
The remaining of this paper is organized as follows. The next section contains a brief description on MOO and VEPSO. In Section 3, the performance measure of the MOO algorithm will be explained. The result and discussion are presented in Section 4. Finally, Section 5 concludes the findings of this paper.

2. Multi-Objective Optimization

2.1. Multi-Objective Optimization Problem

Most real problems involve optimization of more than one objective. Usually, those objectives are conflicting with each other and hence, there will be no single solution exists that satisfies all the objectives. Consider a minimization MOO problem, which has an ***n***-dimensional search space of $x=\{x_1,\cdots,x_n\}$ containing all possible solutions for a ***m***-objective functions of $f(x)=\{f_1(x),\cdots,f_m(x)\}$ that fulfil an ***l***-inequality constraints, $g_i(x)\le 0$, where $i=1,\cdots,l$. The MOO problem is to find a vector, $x^*=\{x_1^*,\cdots,x_n^*\}\in x$ that is optimized for $f(x)$ while satisfying all constraints. The conflicting objectives cause difficulty to obtain a global minimum. As a result, a concept called non-dominated solution is employed to obtain a set of solutions which considers the trade-off among the objectives.
Non-dominated solutions are defined as follows. Given $u=\{u_1,\cdots,u_m\}$ and $v=\{v_1,\cdots,v_m\}$ as two vectors, u *dominates* v if and only if $u_i \le v_i$ for all ***i***-objectives and $u_i < v_i$ for at least one objective. A solution x of MOO problem is a non-dominated solution if and only if there is no other solution x' that has $f(x)$ *dominate* $f(x')$. A set of non-dominated solutions in a search space is usually referred as *Pareto Optimal Set*. While, the set of objective vectors with respect to the *Pareto Optimal Set* is known as the Pareto *Optimal Front* or *Pareto Frontier*.

[1]Universiti Malaysia Pahang, 26600 Pekan, Pahang, Malaysia
** Corresponding Author: Email: zuwairie@ump.edu.my*
 # This paper has been presented at the International Conference on Advanced Technology & Sciences (ICAT'14) held in Antalya (Turkey), August 12-15, 2014.

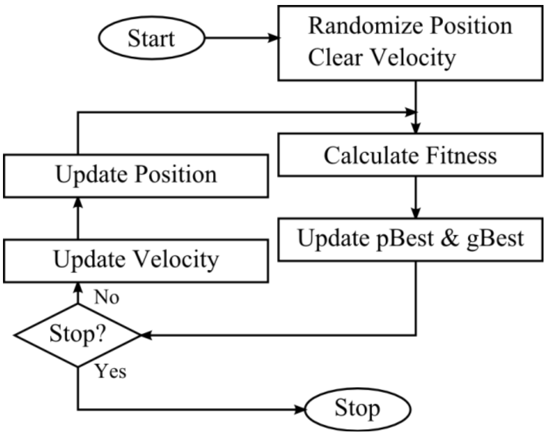

Figure 1. The particle swarm optimization algorithm

2.2. Particle Swarm Optimization

Particle swarm optimization has been introduced by Kennedy and Eberhart [1] for solving SOO problems. PSO algorithm is inspired by the social behavior of bird flocking and fish schooling [20]. In PSO, a swarm of individuals known as particles flies within a search space that contains all the possible solutions. The position of each particle represents the solution for a problem. Each particle in the swarm uses its own and social information to move in the search space.

The PSO algorithm is shown in Fig. 1. During initialization, all particles are randomly positioned in the search space and its velocity is set to zero. Then the particles' fitness is calculated before the particle own and global best positions are updated. The algorithm proceeds by updating the velocity and position based on the Eq. (1) and Eq. (2).

$$v_{in}(t+1) = \omega v_{in}(t) + c_1 r_1 \left(pBest_{in} + p_{in}(t)\right) + c_2 r_2 \left(gBest_{in} + p_{in}(t)\right) \quad (1)$$

$$p_{in}(t+1) = p_{in}(t) + v_{in}(t+1) \quad (2)$$

where ***i*** is the number of particles in an ***n***-dimensional search space. The velocity and position of the particle is denoted as $v_i(t)$ and $p_{in}(t)$, respectively, ω is the weight inertia, and $\boldsymbol{r}_1$ and $\boldsymbol{r}_2$ are random numbers range between zero to one. Besides, the $\boldsymbol{c}_1$ and $\boldsymbol{c}_2$ are the cognitive and social constant, respectively, that determine the influence of the own and social information toward the velocity update. The velocity update also considers two variables, which are the particle own best position, $pBest_{in}(t)$, and the swarm best position, $gBest_{in}(t)$, respectively. After the position is updated, the fitness is calculated again and the particle's own and global best position are updated until the stopping criteria meet.

2.3.Vector-Evaluated Particle Swarm Optimization

The main difference of VEPSO compared to the original PSO algorithm is information sharing. Specifically, the position update in one swarm is influenced by its neighbor swarm's best positions. Thus, the equation for velocity update is modified as follows:

$$v_{\sin}(t+1) = \omega v_{\sin}(t) + c_1 r_1 \left(pBest_{\sin} + p_{in}(t)\right) + c_2 r_2 \left(gBest_{kin} + p_{\sin}(t)\right) \quad (3)$$

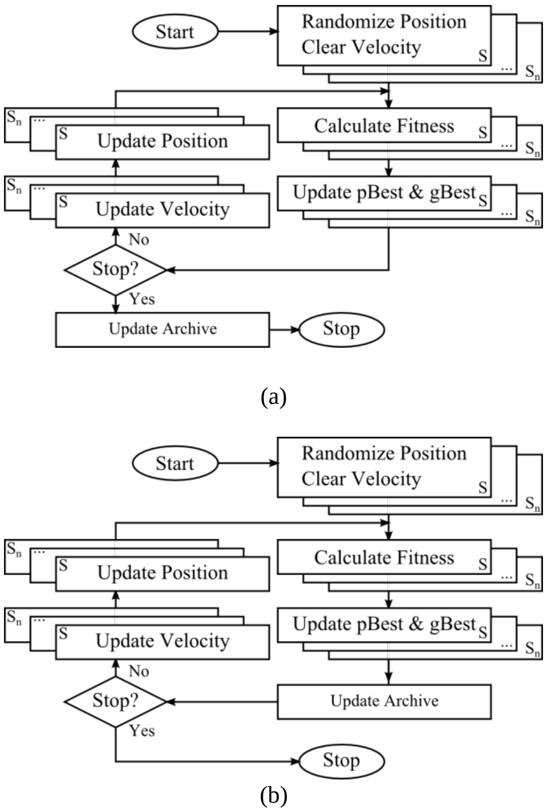

Figure 2. Two possible archive update in VEPSO (a) after an iteration (b) during an iteration

$$k = \begin{cases} 1 & , s = m \\ s-1 & , otherwise \end{cases} \quad (4)$$

where $s = \{1, 2, ..., m\}$ and m is the number of objectives.

Even though each swarm searches the best solution based on its own objective, however, because of the information exchange between swarms, multiple 'trade-off' solutions could be found. Thus, in order to make sure all the solutions satisfies the *Pareto Dominance* concept, a non-dominated selection process is applied in this extended algorithm.

The VEPSO algorithm is shown in Fig. 2. Each step is repeated for all swarm concurrently as in the PSO algorithm. However, there is additional process for selecting non-dominated solutions by update them into an archive. Since Parsopoulos and Vrahatis have not clearly specified the archive update process [8], Fig. 2 shows two possible archive updates in VEPSO algorithm. In Fig. 2(a), the archive is updated after the iterations end [1, 11, 21] such that the non-dominated solutions found is the final solutions from those particles. This algorithm is denoted as "End of Iterations" (*EoI*) in this work. On the other hand, Fig. 2(b) shows that the archive is updated during iterations [22-27]. Hence, this algorithm is denoted as "During Iterations" (*DuI*).

3. Performance Measure And Test Problems For MOO

The first performance measure used in this study is Generalized Distance (*GD*). *GD* is commonly used for measuring algorithm convergence ability [6, 15, 28]. *GD* measures the distance between the obtained *Pareto Optimal Front* (*PF*), $\boldsymbol{PF}_{\text{obtained}}$, and the true *Pareto Optimal Front* (*PF*), $\boldsymbol{PF}_{\text{true}}$. Let the modulus, $\|\cdots\|$, be the count for the element, $(\cdots)$, the *GD* can be formulated as in Eq. (5).

Table 1. VEPSO Parameters

Parameter	Value
Function evaluation	250,000
Number of swarm	2
Number of particle in each swarm	50
Number of iteration	250
Inertia weight, ω	Linearly degrade from 1.0 to 0.4

$$GD = \frac{\left(\sum_{q=1}^{\|PF_{obtain}\|}(d_q)^m\right)^{\frac{1}{m}}}{\|PF_{obtained}\|} \tag{5}$$

where $\boldsymbol{m}$ is the number of objective and d_q is the minimum distance from q-th solution to the $\boldsymbol{PF}_{\text{true}}$.

In order to measure the diversity of non-dominated solutions, a performance measure called *Spread* [15, 28-29] has been considered in this study. *Spread* evaluates the distance difference between all the solutions as follows:

$$Spread = \frac{d_f + d_l + \sum_{q=1}^{\|PF_{obtain}\|-1}\left|d_q - \bar{d}\right|}{d_f + d_l + \left(\|PF_{obtain}\|-1\right)\bar{d}} \tag{6}$$

where d_f and d_l are the Euclidean distance between the extreme solutions in $\boldsymbol{PF}_{\text{obtained}}$ and $\boldsymbol{PF}_{\text{true}}$. When all the solutions in $\boldsymbol{PF}_{\text{true}}$ is arranged in descending, d_q is the distance between one solution to the next solution while $\bar{d}$ is the mean distance for all the solution in the $\boldsymbol{PF}_{\text{obtained}}$. Note that the calculation of *Spread* also includes the extreme solutions from the $\boldsymbol{PF}_{\text{true}}$. The extreme solution is the best solution with respect to one objective but it is also the worst solution with respect to another objective.

Hypervolume (*HV*) [16] measures are also included in this work to evaluate the convergence and diversity performance. *HV* measure the area or volume enclosed by the $\boldsymbol{PF}_{\text{obtained}}$ and a reference point which defined from the worst fitness in the $\boldsymbol{PF}_{\text{true}}$.

In all measures, the obtained *PF* is produced by the VEPSO algorithm. However, the $\boldsymbol{PF}_{\text{true}}$ requires well-defined non-dominated solutions for each MOO problems. Therefore, the $\boldsymbol{PF}_{\text{true}}$ used in this work will be based on the standard database from the jMetal (http://jmetal.sourceforge.net/problems.html).

Regarding the test problems, Zitzler, Deb and Thiele [17] have designed six MOO test problems in which each problem focuses on one kind of problem feature. These problems were abbreviated as ZDT1 to ZDT6. However, in this study, the ZDT5-based evaluation is not considered since the ZDT5 is a binary coded test problem for discrete optimization problems.

In this study, another common test problems called DTLZ [17] is considered as well. The DTLZ, which is abbreviated from Deb, Thiele, Laumanns, and Zitzler, consists of seven MOO test problems in order to extensively evaluate different features of MOO problems.

A disadvantage of ZDT and DTLZ is that both test problems are separable and degenerate [23]. Hence, another test problem called WFG has been proposed by Huband *et al.* [18]. WFG test problem is also chosen in this study.

4. Experiment, Result, and Discussion

The parameters used for the test problems followed the original papers of ZDT [17] and DTLZ [18], whereas for the WFG [19], similar parameters in [30] were used. Meanwhile, the number of objective for all test problems was restricted to two and the VEPSO parameters used is tabulated in Table 1. The experiments for each problem were repeated for 100 runs and then the average convergence and diversity values are calculated.

Table 2 shows the performance of VEPSO algorithm when tested on ZDT test problems. The VEPSO algorithm with different archive updates, namely *EoI* and *DuI*, are analysed with two different settings: $c_1 = c_2 = 0.5$ and $c_1 = c_2 = 1.0$. From this table, the VEPSO algorithm which updates archive during iterations shows better performance in both settings except ZDT2 and in the *NS* for ZDT4. However, for ZDT2, the VEPSO which update at the end of the iterations are much better in *NS* for both setting. This do not directly indicate the *EoI* is a better configuration. Therefore, when $c_1 = c_2 = 0.5$, both *EoI* and *DuI* are inconclusive for which is better as they has almost similar *GD*. However, when $c_1 = c_2 = 1.0$, the *DuI* does results in lower *GD* value which indicates the obtained Pareto front is closer to the true Pareto front, again, but with lower number of solutions found.

Note that the ZDT4 is a difficult problem because there are 2^{29} local optima solutions exist in its search space. However, it is found that *DuI* able to obtain better optima solution since the *GD* obtained is smaller than the *GD* obtained by *EoI*. Additionally, in ZDT4 problem, only the VEPSO algorithm with *DuI* and $c_1 = c_2 = 1.0$ was able to obtained *HV* value.

The performance of VEPSO algorithm when tested on DTLZ test problems is listed in Table 3. In short, the VEPSO algorithm with *DuI* is better at most performance measures. However, when $c_1 = c_2 = 0.5$, the VEPSO algorithm with *EoI* has better *NS* measures than *DuI* in DTLZ1 and DTLZ6. In contrast, extremely large performance difference in convergence could be observed in DTLZ1 and DTLZ3 where these problems also have similar multi local optima solutions feature as in ZDT4. Even the *SP* measure is hardly conclusive; the *HV* measure does indicates that VEPSO algorithm with *DuI* has superiority over algorithm with *EoI*.

The result based on WFG test problems are shown in Table 4. Similarly, the VEPSO algorithm with *DuI* shows better performance than the *EoI*. In addition, as compared to previous test problems, the *DuI* consistently shows better *NS* measures than *EoI*. Again, based on the information of *SP* measure, it was difficult to conclude either *EoI* or *DuI* is better. However, the superiority in *GD* and *HV* measures does imply that the *DuI* has better overall performance.

Generally, based on the results of all test problems, VEPSO algorithm with *DuI* does performs better in two different settings. In order to explain this superiority, first, the movement of the fitness vector (solution objectives) for a particle is observed in the objective space domain for every iteration. Secondly, for better visual analysis, the fitness vectors is filtered using non-dominated selection process to obtain the non-dominated vectors and then, it is labeled with their respective number of iteration, as illustrate in Fig. 3. Besides, the vector for the last iteration of 250 is displayed as well to differentiate with the filtered non-dominated vectors.

In Fig. 3, it is obvious that the vector at the last iteration do not dominate the rest of the vectors. This vector is non-dominated to several other vectors (labeled with 67, 113, 161, 162 and 223). While, the rest filtered solutions (labeled with 123, 134, 150, 187, 196 and 200) were actually dominating the last solution which means they are better than the last solution. However, in VEPSO algorithm with *EoI*, only the last solution is chosen for non-dominated selection whereas the better solutions found during the computation are ignored. Therefore, the VEPSO algorithm with *DuI* preserves good solutions whenever it is found during the computation when the last solutions that may possibly be a worst solution.

Table 3.Performance of VEPSO algorithm when tested on ZDT test problems

Problem	Measures	$c_1 = c_2 = 0.5$		$c_1 = c_2 = 1.0$	
		EoI	***DuI***	***EoI***	***DuI***
ZDT1	*NS*	18.240000	31.950000	19.370000	29.890000
	GD	0.598185	0.358614	0.562556	0.306671
	SP	0.860619	0.888452	0.848210	0.848180
	HV	0.000000	0.000000	0.000000	0.000004
ZDT2	*NS*	8.100000	4.760000	11.090000	7.970000
	GD	1.149853	1.177792	0.854514	0.787217
	SP	0.948850	0.934661	0.909496	0.938282
	HV	0.000000	0.000000	0.000000	0.000000
ZDT3	*NS*	16.120000	39.280000	15.890000	33.340000
	GD	0.380908	0.186525	0.383453	0.175611
	SP	0.827436	0.921501	0.818477	0.883538
	HV	0.000000	0.000103	0.000026	0.000774
ZDT4	*NS*	6.600000	7.450000	7.130000	7.110000
	GD	38.562324	14.823281	33.073706	7.893276
	SP	0.893332	0.939747	0.879965	0.873988
	HV	0.000000	0.000000	0.000000	0.043327
ZDT6	*NS*	9.310000	24.820000	9.520000	46.450000
	GD	2.011100	1.254921	1.631496	0.731704
	SP	0.959094	0.997705	0.910200	1.117636
	HV	0.000000	0.000000	0.011002	0.135575

Table 4. Performance of VEPSO algorithm when tested on DTLZ test problems

Problem	Measures	$c_1 = c_2 = 0.5$		$c_1 = c_2 = 1.0$	
		EoI	***DuI***	***EoI***	***DuI***
DTLZ1	*NS*	9.570000	7.910000	7.270000	5.320000
	GD	101.883042	21.520816	98.211741	14.315669
	SP	0.811414	0.797738	0.779129	0.882947
	HV	0.000000	0.005000	0.002500	0.180723
DTLZ2	*NS*	17.760000	95.350000	15.990000	73.090000
	GD	0.046790	0.006503	0.078892	0.004824
	SP	0.730134	0.832162	0.769422	0.793372
	HV	0.083728	0.143988	0.093191	0.167805
DTLZ3	*NS*	7.260000	10.940000	6.090000	7.620000
	GD	243.370239	91.918356	198.283297	55.412570
	SP	0.879851	0.919220	0.905444	0.898635
	HV	0.000000	0.000000	0.000000	0.011662
DTLZ4	*NS*	6.950000	23.870000	5.590000	21.850000
	GD	0.099813	0.028737	0.111334	0.025031
	SP	1.058371	1.074169	1.032273	1.167165
	HV	0.001586	0.062433	0.002456	0.062361
DTLZ5	*NS*	17.240000	96.480000	16.360000	74.960000
	GD	0.045383	0.006751	0.073864	0.004971
	SP	0.701103	0.850762	0.754325	0.779270
	HV	0.080995	0.140817	0.096673	0.166771
DTLZ6	*NS*	15.280000	9.830000	5.520000	6.730000
	GD	4.513450	4.006382	6.178312	3.739334
	SP	0.917620	0.817805	0.862215	0.833595
	HV	0.000000	0.000000	0.000000	0.000000
DTLZ7	*NS*	9.500000	10.590000	11.580000	14.890000
	GD	1.132059	0.770572	0.901702	0.479538
	SP	0.905583	0.919789	0.876098	0.898720
	HV	0.000000	0.000000	0.000000	0.000035

Table 5. Performance of VEPSO algorithm when tested on WFG test problems

Problem	Measures	$c_1 = c_2 = 0.5$		$c_1 = c_2 = 1.0$	
		EoI	*DuI*	*EoI*	*DuI*
WFG1	*NS*	14.370000	99.940000	19.640000	99.980000
	GD	0.132771	0.047608	0.113948	0.047640
	SP	0.990042	0.957187	1.306888	0.843295
	HV	0.000000	0.100283	0.023606	0.104358
WFG2	*NS*	10.340000	51.730000	12.870000	41.510000
	GD	0.046071	0.019043	0.041782	0.019344
	SP	0.803569	1.088650	0.779051	1.001560
	HV	0.341787	0.400725	0.367339	0.424419
WFG3	*NS*	28.100000	99.940000	33.930000	99.830000
	GD	0.022213	0.009526	0.015977	0.007256
	SP	0.722680	0.699150	0.693586	0.559141
	HV	0.298578	0.336630	0.336536	0.365673
WFG4	*NS*	32.360000	98.180000	37.600000	95.850000
	GD	0.019747	0.008262	0.014500	0.006784
	SP	0.712605	0.686858	0.716378	0.604025
	HV	0.098504	0.130089	0.115638	0.145196
WFG5	*NS*	25.130000	44.530000	21.800000	54.670000
	GD	0.050704	0.025509	0.044493	0.014933
	SP	0.707043	0.665721	0.682074	0.663076
	HV	0.038964	0.084897	0.060983	0.132675
WFG6	*NS*	23.850000	90.770000	16.940000	46.780000
	GD	0.045762	0.017541	0.044006	0.015198
	SP	0.701535	0.804844	0.721396	0.842422
	HV	0.045158	0.074543	0.051280	0.118054
WFG7	*NS*	33.100000	99.990000	39.620000	99.780000
	GD	0.025956	0.012239	0.021883	0.010647
	SP	0.669851	0.544235	0.664195	0.547616
	HV	0.073608	0.095537	0.081870	0.106423
WFG8	*NS*	28.490000	99.260000	30.390000	98.300000
	GD	0.046024	0.019130	0.039781	0.016889
	SP	0.717970	0.716757	0.667089	0.659762
	HV	0.055402	0.078007	0.061782	0.085734
WFG9	*NS*	18.010000	75.800000	15.370000	86.320000
	GD	0.045011	0.011732	0.035357	0.007480
	SP	0.645567	0.699325	0.623683	0.571291
	HV	0.079966	0.143781	0.095532	0.163231

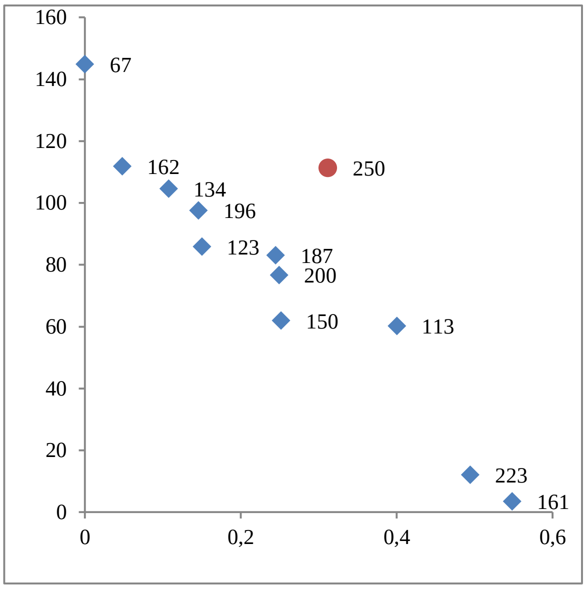

Figure 3. Fitness vectors that dominate the fitness vector at last iteration

5. Conclusions

In this study, two possible archive updates for VEPSO algorithm are analysed for its convergence and diversity performance based on quantitative evaluation using several performance measures. From the results, the VEPSO algorithm which updates its archive after the end of iterations has difficulty to produce good non-dominated solutions. However, the VEPSO algorithm which updates its archives during iterations does show better overall performance. This is due to the fitness vector found at the end of iteration is actually worse than those found during the iterations.

Acknowledgements

This work is supported by the Research Acculturation Grant Scheme (RDU121403) and MyPhD Scholarship from Ministry of Higher Education of Malaysia.

References

[1] J. Kennedy and R. Eberhart. Particle swarm optimization. in Proceedings IEEE International Conference on Neural Networks. 1995, volume 4, pages 1942-1948.

[2] D. Besozzi, P. Cazzaniga, G. Mauri, D. Pescini, and L. Vanneschi, A Comparison of Genetic Algorithms and Particle Swarm Optimization for Parameter Estimation in Stochastic Biochemical Systems, in Evolutionary Computation, Machine Learning and Data Mining in Bioinformatics, C. Pizzuti, M. Ritchie, and M. Giacobini, Editors. 2009, Springer Berlin / Heidelberg. p. 116-127.

[3] R. Hassan, B. Cohanim, O. De Weck, and G. Venter. A Comparison Of Particle Swarm Optimization And The Genetic Algorithm. in AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference. 2005, volume pages 1-13.

[4] X. Hu and R. Eberhart. Multiobjective optimization using dynamic neighborhood particle swarm optimization. in Congress on Evolutionary Computation (CEC 2002). 2002, volume 2, pages 1677-1681. IEEE Computer Society.

[5] C. A. Coello Coello and M. S. Lechuga. MOPSO: a proposal for multiple objective particle swarm optimization. in Congress on Evolutionary Computation (CEC 2002). 2002, volume 2, pages 1051-1056.

[6] G. T. Pulido and C. A. Coello Coello, Using Clustering Techniques to Improve the Performance of a Multi-objective Particle Swarm Optimizer, in Genetic and Evolutionary Computation. 2004, Springer Berlin / Heidelberg. p. 225-237.

[7] K. E. Parsopoulos and M. N. Vrahatis. Particle swarm optimization method in multiobjective problems. in Proceedings of the ACM symposium on Applied computing. 2002, volume pages 603-607. Madrid, Spain: ACM.

[8] J. D. Schaffer, Some experiments in machine learning using vector evaluated genetic algorithms (artificial intelligence, optimization, adaptation, pattern recognition), in Faculty Of Graduate School. 1984, Vanderbilt University: Nashville, Tennessee. p. 166.

[9] S. M. V. Rao and G. Jagadeesh, Vector Evaluated Particle Swarm Optimization of Supersonic Ejector for Hydrogen Fuel Cells. Journal of Fuel Cell Science and Technology, 2010. 7(4): p. 041014-7.

[10] D. Gies and Y. Rahmat-Samii. Vector evaluated particle swarm optimization: optimization of a radiometer array antenna. in IEEE Antennas and Propagation Society International Symposium. 2004, volume 3, pages 2297-2300.

[11] S. N. Omkar, D. Mudigere, G. N. Naik, and S. Gopalakrishnan, Vector evaluated particle swarm optimization (VEPSO) for multi-objective design optimization of composite structures. Computers and Structures, 2008. 86(1-2): p. 1-14.

[12] Z. Ibrahim, N. K. Khalid, J. A. Ahmed Mukred, S. Buyamin, Z. Md Yusof, M. F. Mohamed Saaid, Norrima Mokhtar, and A. P. Engelbrecht, A DNA sequence design for DNA computation based on binary vector evaluated particle swarm optimization. International Journal of Unconventional Computing, 2012. 8(2): p. 119-137.

[13] J. Grobler, Particle swarm optimization and differential evolution for multi objective multiple machine scheduling, in Department of Industrial and Systems Engineering 2009, University of Pretoria: Pretoria, South Africa. p. 159.

[14] D. A. Van Veldhuizen, Multiobjective evolutionary algorithms: classifications, analyses, and new innovations, in Faculty of the Graduate School of Engineering. 1999, Air Force Institute of Technology, Air University. p. 249.

[15] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions On Evolutionary Computation, 2002. 6(2): p. 182-197.

[16] E. Zitzler and L. Thiele, Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. IEEE Transactions On Evolutionary Computation, 1999. 3(4): p. 257-271.

[17] E. Zitzler, K. Deb, and L. Thiele, Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. Evolutionary Computation, 2000. 8(2): p. 173-195.

[18] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, Scalable Test Problems for Evolutionary Multi-Objective Optimization, in KanGAL Report 2001001. 2001, Kanpur Genetic Algorithms Laboratory, Indian Institute of Technology Kanpur: Kanpur, India. p. 27.

[19] S. Huband, L. Barone, L. While, and P. Hingston, A Scalable Multi-objective Test Problem Toolkit, in Evolutionary Multi-Criterion Optimization, C.A. Coello Coello, A. Hernández Aguirre, and E. Zitzler, Editors. 2005, Springer Berlin / Heidelberg. p. 280-295.

[20] J. Kennedy, R. C. Eberhart, and Y. Shi, Swarm Intelligence. The Morgan Kaufmann Series in Evolutionary Computation, ed. D.B. Fogel. 2001, San Francisco: Morgan Kaufmann Publishers. 512.

[21] N. K. Khalid, Particle Swarm Optimization for Solving DNA Sequence Design Problem, in Faculty of Electrical Engineering. 2010, Universiti Teknologi Malaysia: Skudai, Johor Bahru. p. 148.

[22] M. A. Abido. Two-level of nondominated solutions approach to multiobjective particle swarm optimization. in Proceedings of the Conference on Genetic and Evolutionary Computation. 2007, volume pages 726-733. London, England: ACM.

[23] J. E. Alvarez-Benitez, R. M. Everson, and J. E. Fieldsend, A MOPSO Algorithm Based Exclusively on Pareto Dominance Concepts, in Evolutionary Multi-Criterion Optimization, C.A. Coello Coello, A. Hernández Aguirre, and E. Zitzler, Editors. 2005, Springer Berlin / Heidelberg. p. 459-473.

[24] J. E. Fieldsend and S. Singh. A multi-objective algorithm based upon particle swarm optimization, an efficient data structure and turbulence. in Workshop on Computational Intelligence. 2002, volume pages 37–44.

[25] S. Mostaghim and J. Teich. Strategies for finding good local guides in multi-objective particle swarm optimization (MOPSO). in IEEE Proceedings of the Swarm Intelligence Symposium (SIS 2003). 2003, volume pages 26-33.

[26] C. A. Coello Coello, G. T. Pulido, and M. S. Lechuga, Handling multiple objectives with particle swarm optimization. IEEE Transactions On Evolutionary Computation, 2004. 8(3): p. 256-279.

[27] X. Hu, R. C. Eberhart, and Y. Shi. Particle swarm with extended memory for multiobjective optimization. in IEEE Swarm Intelligence Symposium 2003. 2003, volume pages 193. Indianapolis, IN, USA: IEEE.

[28] S.-K. S. Fan and J.-M. Chang, A parallel particle swarm

optimization algorithm for multi-objective optimization problems. Engineering Optimization, 2009. 41(7): p. 673 - 697.

[29] J. Durillo, J. García-Nieto, A. Nebro, C. A. Coello Coello, F. Luna, and E. Alba, Multi-Objective Particle Swarm Optimizers: An Experimental Comparison, in Evolutionary Multi-Criterion Optimization, M. Ehrgott, et al., Editors. 2009, Springer Berlin / Heidelberg. p. 495-509.

[30] S. Huband, P. Hingston, L. Barone, and L. While, A review of multiobjective test problems and a scalable test problem toolkit. IEEE Transactions On Evolutionary Computation, 2006. 10(5): p. 477-506.