Solving Dynamic Multi-Objective Problems with Vector Evaluated Particle Swarm Optimisation

Mardé Greeff, Andries. P. Engelbrecht

***Abstract*— Many optimisation problems are multi-objective and change dynamically. Many methods use a weighted average approach to the multiple objectives. This paper introduces the usage of the vector evaluated particle swarm optimiser (VEPSO) to solve dynamic multi-objective optimisation problems. Every objective is solved by one swarm and the swarms share knowledge amongst each other about the objective that it is solving. Not much work has been done on using this approach in dynamic environments. This paper discusses this approach as well as the effect of the population size and the response methods to a detected change on the performance of the algorithm. The results showed that more non-dominated solutions, as well as more uniformly distributed solutions, are found when all swarms are re-intialised when a change is detected, instead of only the swarm(s) optimising the specific objective function(s) that has changed. Furthermore, an increase in population size results in a higher number of non-dominated solutions found, but can lead to solutions that are less uniformly distributed.**

I. Introduction

Optimisation problems occur in many situations and aspects of modern life. In reality, many of these problems are dynamic in nature, where changes can occur in the environment that influence the solutions of the optimisation problem. In a world where many people travel frequently, the scheduling of aircrafts' take-off and landing, requires careful planning, since sudden changes to the schedule can be caused by an airplane arriving earlier or later and flights being delayed. Two goals of air traffic control can be defined as minimising the waiting time for an aircraft to take-off or land and minimising the possibility of collisions. These two goals are in conflict with one another. In order to decrease the possibility of a collision, the waiting time of an aircraft to take-off or land will be increased. This is just one example where the optimisation problem is a dynamic multi-objective problem (DMOOP).

Generally a DMOOP does not have a single solution. In many cases the objectives are in conflict with one another, and an improvement in one objective leads to a worse solution for at least one of the other objectives. Therefore, the set of solutions that can be found where no other solution is better for all the objectives, is called the Pareto optimal front (POF), and the solutions are called non-dominated solutions.

To efficiently solve a DMOOP the algorithm should be able to track the changing POF efficiently. Therefore, the algorithm should be able to detect that a change has occurred and should be able to respond to the change accordingly. However, the comparison of one algorithm's performance against another in terms of solving a DMOOP, is not a trivial task, since the true POF are unknown in many cases.

This paper proposes using the Vector Evaluated Particle Swarm Optimisation (VEPSO) algorithm to solve DMOOPs and discusses the performance of the VEPSO algorithm with regards to specific DMOO benchmark functions. The effect of the swarm sizes, as well as the effect of various responses to detected changes in the environment, are also highlighted.

The rest of the paper's layout is as follows: Section II provides background information and highlights related work that is relevant for the research discussed in this paper. Section III provides an overview of the VEPSO algorithm and the changes that has been made to the algorithm for DMOOPs. The benchmark functions and performance metrics that have been used to test the algorithm's performance are discussed in Sections IV and V respectively. Section VI describes the experiments that have been done to test the algorithm's performance and the results of the experiments are highlighted in Section VII. Finally, Section VIII discusses conclusions on the work presented in this paper.

II. Background

Particle Swarm Optimisation (PSO), introduced by Eberhart and Kennedy [1], is a population-based optimisation method inspired by the social behaviour of bird flocks. Each PSO swarm consists of a number of particles that move in the search space in search of solutions. Each individual particle has a current position in the search space, $\mathbf{x}_i$, a current velocity, $\mathbf{v}_i$, and a personal best position in the search space, $\mathbf{y}_i$, where the particle had the smallest error with regards to the objective function. The position amongst all the particles' personal best positions that yielded the smallest error, is called the global best position, denoted as $\hat{\mathbf{y}}$. During each iteration every particle's velocity is updated and the new velocity is added to the particle's current position to determine its new position.

PSO has been successfully applied to solve dynamic single-objective optimization problems ([2], [3], [4], [5]). However, when dealing with dynamic problems, it is vital that the algorithm can detect that a change has occurred and then respond to the change in an appropriate manner. Carlisle and Dozier [6] introduced the concept of a sentry particle to detect whether a change has occurred in the environment. A sentry particle is randomly selected after each iteration, and then re-evaluated before the next iteration. During the re-evaluation its current fitness value is compared with its previous fitness value, i.e. its fitness value after the previous iteration. If the two values differ more than a specified value,

Mardé Greeff is with the Meraka Institute, CSIR, South Africa (email: mgreeff@csir.co.za). Andries Engelbrecht is with the Computer Science Department, University of Pretoria, South Africa (email: engel@cs.up.ac.za).

978-1-4244-1823-7/08/$25.00©2008 IEEE

the swarm is alerted that a change has taken place and responds accordingly. Hu and Eberhart [7] suggested to use the global best, and global second best particles as sentries.

Once a change in the environment has been detected, the algorithm should respond effectively to these changes. One approach is to re-calculate the value of each particle's personal best [8]. If the new value is less fit than the particles's current position, its personal best value is replaced by its current position. This comparison ensures that valid past experience is not lost. Another approach is to re-initialize a percentage of the swarm population. This ensures that the swarm does not remain in a small area after the change has occurred and a portion of the particles do not lose their memory, which could be valuable information if the change is small. Cui *et al.* [5] proposed the usage of an evaporation constant, that can take a value between zero and one, to update the particle's best fitness value. This will cause the particle's memory to gradually decrease over time, until at a certain point in time the particle's current fitness will be better than the decreased fitness value. When this happens, the decreased fitness value will be replaced by the particle's current fitness. With this approach the evironment is not monitored by any particles, as is the case with the usage of sentry particles.

More recently research has been done on using evolutionary algorithms (EAs) ([9], [10], [11]) and PSO ([12], [13]) to solve DMOOPs. In order to test and analyse an algorithm's capability of tracking a dynamic Pareto optimal front, benchmark functions are used. Jin and Sendhof [14] introduced a way to define a dynamic two-objective optimisation problem by reformulating a three-objective optimisation test function. Guan *et al.* [9] created dynamic problems by replacing objectives with new ones at specific times. Many benchmark functions were developed by adapting static MOO benchmark functions to dynamic ones. Farina *et al.* [15] developed a number of test functions for DMOO based on the static MOO two-objective ZDT functions [16] and the scalable DTLZ functions [17]. Some adaptions to these test functions were proposed in ([10], [18]). Mehnen *et al.* [10] also proposed DSW functions that are adapted from the static MOO function problem of Schaffer [19]. Others added noise to Deb's functions ([20], [21]).

In order to compare one algorithm's performance against another algorithm, performance metrics are required. For DMOOP two groups of performance metrics exist, namely performance metrics where the true POF is known and performance metrics where the true POF is unknown. For a known POF the convergence, measured by the generational distance proposed by Van Veldhuizen [22], and spread or distribution of the solutions are often used to measure an algorithm's performance ([21], [23]). Branke *et al.* [12] proposed the reversed generational distance and the collective mean error as performance metrics. Another metric is the $HVR(t)$ metric, which represents the ratio of the hypervolume of the solutions and the hypervolume of the known POF at a specific time ([22], [12]). Li *et al.* [12] proposed a metric of spacing that can be used when the true POF is unknown. The size of the non-dominated solution set can also be used to measure an algorithm's performance [18]. Cámara *et al.* proposed measures of accuracy, stability and reaction capacity of an algorithm, that are based on the calculation of the hypervolume of the non-dominated solution set [13].

III. VEPSO

Parsopoulos *et al.* [24] introduced the Vector Evaluated Particle Swarm Optimisation (VEPSO) that is a multi-swarm variation of PSO, inspired by the Vector Evaluated Genetic Algorithm (VEGA) [25]. In VEPSO each swarm solves one objective function and shares its knowledge with the other swarms. The shared knowledge is used to update the velocity of the particles as indicated in Equations (1) and (2) below:

$$v_i^j(t+1) = \chi^j[w^j v_i^j(t) + c_1^j r_1(y_i^j(t) - x_i^j(t)) + c_2^j r_2(\hat{y}_i^s(t) - x_i^j(t))] \tag{1}$$

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

where n represents the dimension with $i = 1, \ldots, n$; m represents the number of swarms with $j = 1, \ldots, m$ as the swarm index; $\hat{y}_i^s$ is the global best of the s-th swarm; c_1^j and c_2^j are the cognitive and social parameters of the j-th swarm respectively; $r_1, r_2 \in [0, 1]$; w^j and χ^j are the inertia weight and constriction factor of the j-th swarm respectively; and $s \in [1, \ldots, j-1, j+1, \ldots, M]$ represents the index of a respective swarm.

In Equation (1) the global best of another swarm (indexed by s) is used to update the velocity of the particles of the j-th swarm. The index s can be set up in various ways, affecting the topology of the swarms in VEPSO, e.g. if s is selected according to Equation (3) the swarms will have a ring topology, where

$$s = \begin{cases} M & \text{for } j = 1 \\ j-1 & \text{for } j = 2, \ldots, M \end{cases} \tag{3}$$

Sentry particles are used to detect a change [6] in the environment. After each iteration a sentry particle is randomly selected and re-evaluated before the next iteration. If the two fitness values differ more than a specified value, the swarm is alerted that a change has taken place and responds accordingly, as explained in the pseudo code below:

```
for number of iterations do
    check whether change occurred
    if change occurred
        respond to change
        re-evaluate solutions in POF
        remove dominated solutions from POF
    perform iteration
    if new solutions are non-dominated
        if space in archive
            add new solutions to POF
        else
            remove solutions from archive
            add new solutions to archive
    select sentries
```

IV. Benchmark functions

To test the performance of VEPSO with DMOOPs, with two or more objective functions, as well as various types of POFs, two functions proposed by Farina *et al.* [15] are used. Furthermore, two functions introduced by Abido [26] for static MOOPs, are modified to create DMOOPs.

Below, τ is the generation counter, τ_t is the number of iterations for which t remains fixed and n_t is the number of distinct steps in t.

The FDA1 problem is depticted as:

$$\mathsf{FDA1} = \begin{cases} Minimize: f(\mathbf{x},t) = (f_1(\mathsf{x_I},t), g(\mathbf{x_{II}},t)\cdot \\ \qquad h(\mathbf{x_{III}}, f_i(\mathsf{x_I},t), g(\mathbf{x_{II}},t),t)) \\ \\ f_1(\mathsf{x_I}) = x_i \\ g(\mathbf{x_{II}}) = 1 + \sum_{x_i \in \mathbf{x_{II}}} (x_i - G(t))^2 \\ h(f_1, g) = 1 - \sqrt{\frac{f_1}{g}} \\ where: \\ G(t) = \sin(0.5\pi t), \;\; t = \frac{1}{n_t}\left\lfloor \frac{\tau}{\tau_t} \right\rfloor \\ \mathsf{x_I} \in [0,1]; \;\; \mathbf{x_{II}} = (x_2, \ldots, x_n) \in [-1,1] \end{cases} \tag{4}$$

The function parameters were set as $n = 20$ and $n_t = 10$ (as suggested by [15]). Function FDA1 (Equation (4)) has a convex POF where the values in the decision variable space changes, but the values in the objective space remains the same.

The FDA4 problem:

$$\mathsf{FDA4} = \begin{cases} Minimize: f_1(\mathbf{x}), \ldots, f_M(\mathbf{x}) \\ min_x: f_1(\mathbf{x}) = (1 + g(\mathbf{x_{II}})) \prod_{i=1}^{M-1} \cos(\frac{x_i \pi}{2}) \\ min_x: f_k(\mathbf{x}) = (1 + g(\mathbf{x_{II}}))(\prod_{i=1}^{M-k} \cos(\frac{x_i \pi}{2})) \\ \sin(\frac{x_{M-k+1}\pi}{2}), k = 2, \ldots, M-1 \\ min_x: f_M(\mathbf{x}) = (1 + g(\mathbf{x_{II}})) \sin(\frac{x_1 \pi}{2}) \\ where: \\ g(\mathbf{x_{II}}) = \sum_{x_i \in \mathbf{x_{II}}} (x_i - G(t))^2 \\ G(t) = |sin(0.5\pi t)| \\ F(t) = 10^{2\sin(0.5\pi t)}, \;\; t = \frac{1}{n_t}\left\lfloor \frac{\tau}{\tau_t} \right\rfloor \\ \mathbf{x_{II}} = (x_M, \ldots, x_n); x_i \in [0,1], \forall i = 1, \ldots, n \end{cases} \tag{5}$$

For FDA4 the parameters were set as $n = M + 9$, $|\mathsf{X_{II}}| = 10$, $n_t = 10$ (as suggested by [15]) and $M = 3$. Function FDA4 (Equation (5)) has a non-convex shaped POF where the values in the decision variable space changes, but the values in the objective space remains the same.

The TP1$_{mod}$ problem:

$$\mathsf{TP1_{mod}} = \begin{cases} Minimize: (f_1(x), f_2(x)) \\ f_1(x) = \begin{cases} -x & \text{for } x \leq 1 \\ -2 + x & \text{for } 1 < x \leq 3 \\ 4 - x & \text{for } 3 < x \leq 4 \\ -4 + x & \text{for } x > 4 \end{cases} \\ f_2(x) = (x-5)^2 + G(t) \\ where: \\ G(t) = |sin(0.5\pi t)|, \;\; t = \frac{1}{n_t}\left\lfloor \frac{\tau}{\tau_t} \right\rfloor \\ -100 \leq x \leq 100 \end{cases} \tag{6}$$

For Function TP1$_{\mathsf{mod}}$ the parameter n_t was set to 10 as suggested by [15] for the FDA functions. Function TP1$_{\mathsf{mod}}$ (Equation (6)) has a discontinuous POF.

The TP2$_{mod}$ problem:

$$\mathsf{TP2_{mod}} = \begin{cases} Minimize: (f_1(x), f_2(x)) \\ f_1(x_1, x_2) = 2 + (x_2 - 1)^2 - 10c_1 G(t) \\ \qquad + (x_1 - 2)^2 \\ f_2(x_1, x_2) = 9x_1 + (x_2 - 1)^2 - 10c_2 G(t) \\ where: \\ c_1 = \begin{cases} c_1 & \text{for } c_1 \leq 0 \\ 0 & \text{for } c_1 > 0 \end{cases} \\ c_2 = \begin{cases} c_2 & \text{for } c_2 \leq 0 \\ 0 & \text{for } c_2 > 0 \end{cases} \\ G(t) = |sin(0.5\pi t)|, \;\; t = \frac{1}{n_t}\left\lfloor \frac{\tau}{\tau_t} \right\rfloor \\ x_1, x_2 \in [-20, 20] \end{cases} \tag{7}$$

For TP2$_{\mathsf{mod}}$ the parameter n_t was set to 10 as suggested by [15] for the FDA functions. Function TP2$_{\mathsf{mod}}$ (Equation (7)) has a convex POF.

V. Performance Metrics

This paper assumes that the POF of the benchmark functions are unknown, since in reality this will often be the case. The performance metrics that are used to compare the performance of various algorithms are discussed below.

A. Number of non-dominated solutions found

The first performance metric is the average number of non-dominated solutions found at each iteration before a change has occurred in the environment. To compare the performance of one algorithm against another, the number of non-dominated solutions found (from now on referred to as the NS metric) is calculated for each iteration before a change in the environment occurs (refer to Equation (8)).

$$NS^i = ns_i, \;\; \overline{NS}^i = \frac{1}{n_r} \sum_{j=1}^{n_r} NS_j^i \tag{8}$$

where n_r is the number of runs, ns_i is the numnber of non-dominated solutions found for iteration i and NS_j^i is the NS metric value of run j at iteration i, which is an iteration before a change occurs in the environment.

The average over 30 runs is then calculated for each of these iterations. The number of times that one algorithm has a better NS average than the others, is counted and referred to as NS_{wins}.

B. Spacing

The metric of spacing [21] indicates how evenly the non-dominated solutions are distributed along the discovered POF, and is defined as

$$S = \frac{1}{n_{PF}} \left[\frac{1}{n_{PF}} \sum_{i=1}^{n_{PF}} (d_i - \overline{d})^2 \right]^{\frac{1}{2}}, \; \overline{d} = \frac{1}{n_{PF}} \sum_{i=1}^{n_{PF}} d_i \tag{9}$$

where n_{PF} is the number of non-dominated solutions found and d_i is the euclidean distance, in the objective space, between non-dominated solution i and its nearest non-dominated solution.

To compare one algorithm against another, the average spacing metric $\overline{S}^i$ is calculated for each iteration before a change in the environment occurs (refer to Equation (9)).

$$\overline{S}^i = \frac{1}{n_r} \sum_{j=1}^{n_r} S_j^i \tag{10}$$

where n_r is the number of runs and S_j^i is the spacing metric value of run j at iteration i, which is an iteration before a change occurs in the environment.

The number of times that one algorithm has a better S average than the others, is counted and referred to as S_{wins}.

C. Hypervolume

The hypervolume (HV) or S-metric [16] computes the size of the region dominated by a set of non-dominated solutions based on a reference vector. The reference vector is constructed using the worst objective values of each objective.

In order to compare one algorithm against another, the HV metric is calculated for each iteration before a change in the environment occurs. The average over 30 runs is then calculated for each of these iterations. The number of times that one algorithm has a better HV average than the others, is counted and referred to as HV_{wins}.

If it is unknown when a change will occur, the performance metrics can be calculated over all iterations instead of the iterations just before a change occurrs.

To determine the algorithm with the best performance for a specific function, its overall rank is calculated. For each of the performance metrics the algorithm is ranked according to its performance with regards to the specific metric. The algorithm's average rank value is calculated and then the algorithm is ranked accordingly.

VI. Experiments

This section describes the experiments that were conducted to test the performance of VEPSO when solving DMOOPs. The benchmark functions and performanc metrics discussed in Sections IV and V respectively were used to test the performance of variations of the VEPSO algorithm.

To test the influence of population size on the performance of the VEPSO algorithm, the number of particles of the swarms was varied between 2, 5, 10, 20, 30 and 40 particles respectively. For each of these population sizes either all swarms' global best particle and second global best particle were re-initialised when a change was detected, or only the forementioned particles of the swarm that is solving the objective function that changed.

To test the effect of the percentage of the swarm's population that is re-initialised when a change is detected in the environment, a fixed population size of 40 was used and either the global best particle and global second best particle, or 10%, 20% or 30% of the population was re-initialised. This was done for either all the swarms or only the swarm that is solving the objective function that has changed.

All experiments consisted of 30 runs and each run consisted of 1 000 iterations. The frequency of change, τ_t, was set to 5 as suggested by [15]. Therefore, during each run the functions change every 5 iterations, resulting in 200 changes per run. Therefore the maximum number of S_{wins}, NS_{wins} and HV_{wins} are 200 per algorithm, since more than one algorithm can have the best values for the specific time. The PSO parameters were set to the following values: w=0.72 and $c1 = c2 = 1.49$, since these values lead to convergent behaviour [27].

VII. Results

This section discusses the results that were obtained from the experiments, with regards to the overall rank of the variations of VEPSO and the effect of the population size on VEPSO's performance. Furthermore, the effect of the response to a detected change in the environment is highlighted. The results of the experiments can be seen in Tables I - VIII. In the Tables (A) indicates that all swarms are re-initialised in response to a change and (C) indicates that only the swarms that solve the objective functions that have changed are re-initialised.

A. Overall Rank of Variations of VEPSO

Considering the overall rank of the variations of VEPSO (as discussed in Section V) a population size of 40 particles where all swarms are re-initialised when a change in the environment is detected, ranked overall mostly in first place (refer to Tables I, II, III and IV).

B. Effect of Population Size

From the results it can be seen that an increase in the population size leads to a higher number of non-dominated solutions found. In some cases this can lead to a higher value of the spacing metric, i.e. the non-dominated solutions are less uniformly distributed.

Figure 1 illustrates various non-dominated solution sets found over various runs with different population sizes for Function FDA1 on the left. On the right only the non-dominated solutions from these non-dominated solution sets are illustrated. Non-dominated solutions found over various runs with various population sizes for Functions TP2$_{mod}$ and FDA4 are illustrated left and right in Figure 2 respectively.

C. Effect of Various Responses to Change

Tables V to VIII highlight the results that were obtained with various variations of response to a detected change. It is interesting to note that re-initialising either the global gbest and second gbest or 10% of the population of all swarms when a change is detected, consistently lead to one of the best 3 performances.

Furthermore, when all swarms are re-initialised in response to a change (indicated by (A) in the Tables), better results are obtained than in the case where only the specific swarm that optimises the objective function that changed is re-initialised (indicated by (C) in the Tables). Re-initialising all swarms when a change is detected, leads to a lower spacing metric value (more uniformly distributed solutions) and a higher number of non-dominated solutions found.

D. Comparison with Other Work

Zhen [18] found the average hypervolume for Function FDA1 as ± 0.6654 and the average hypervolume for Function FDA4 as ± 0.466 when using a GA algorithm. However, it should be noted that for these experiments τ_t was set to

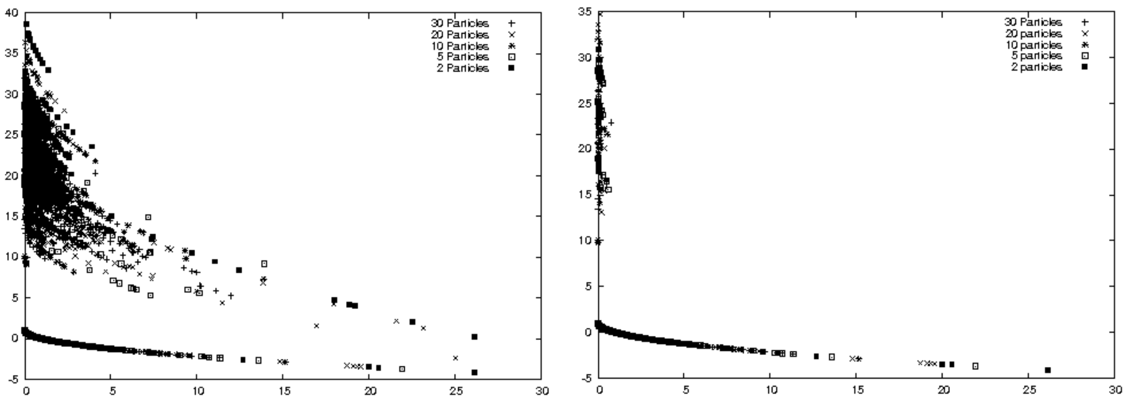

Fig. 1. Solutions found with various population sizes for FDA1

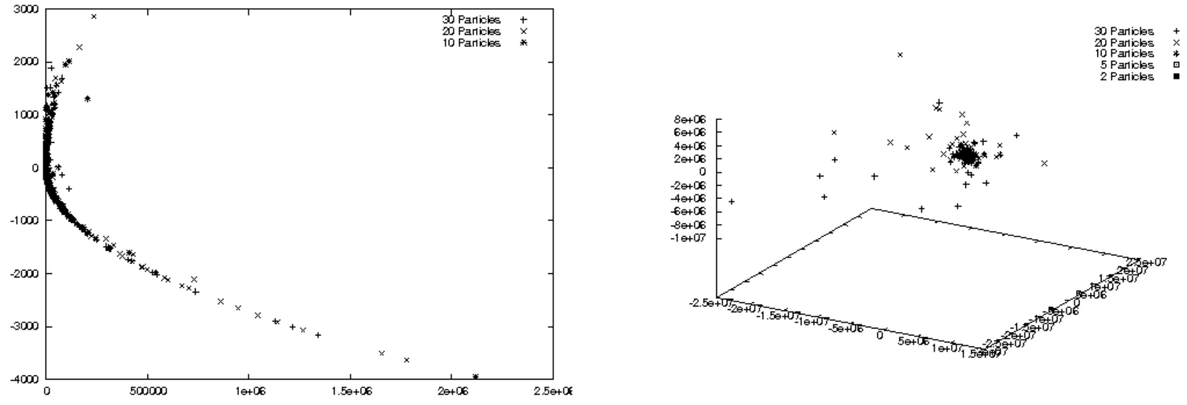

Fig. 2. Non-dominated solutions found with various population sizes for TP2$_{mod}$ on the left and FDA4 on the right

2000, i.e. a change occurred in the environment after 2000 generations.

In the experiments of this paper τ_t was set to 5, i.e. a change occurred every 5 generations. Using VEPSO the highest hypervolume value for Function FDA1 was 1.05 with a population size of 40 and re-initialising 30% of the population of the swarm that optimises the objective function that has changed (refer to Table V). The second highest hypervolume value for FDA1 was 0.57 for a VEPSO algorithm with population size of 20 particles and only the swarm optimising the objective function that has changed are re-initialised (refer to Table I). For Function FDA4 very large hypervolume values were obtained as can be seen in Tables I and V. From the spacing metric values in these tables can be seen that the non-dominated solutions are not very uniformly distributed. Furthermore, from Figure 2 it can be seen that there are a few solutions far away from the others. If one of these values become part of the reference vector in the calculation of the hypervolume, it can explain the high hypervolume values.

VIII. Conclusions

This paper introduced using the VEPSO algorithm for DMOOPs. Through the usage of benchmark functions and performance metrics variations of the VEPSO algorithm were compared with one another.

The effect of the swarms' population size, as well as various responses to changes in the environment were discussed.

The results showed that VEPSO can solve a DMOOP with a discontinuous POF and functions with more than two objectives. An increase in the population size of the swarms lead to more non-dominated solutions found. Furthermore, re-initialising all swarms and not only the swarm(s) optimising the objective function(s) that have changed, resulted in more non-dominated solutions, as well as more uniformly distributed solutions.

Future work will include investigating various performance metrics for DMOOPs, especially for cases where the true POF is unknown.

TABLE I

Spacing, NS and Hypervolume Metric Values for Function FDA1

#Particles	**Performance Metric**						**Rank**
	$\overline{S}$	S_{wins}	$\overline{NS}$	NS_{wins}	$\overline{HV}$	HV_{wins}	
2 (A)	0.19	61	34.45	0	0.53	1	3
2 (C)	0.33	2	12.32	0	2.67	198	1
5 (A)	0.19	137	40.79	2	0.47	1	6
5 (C)	0.45	2	13.34	0	0.25	0	5
10 (A)	0.23	0	36.0	0	0.17	0	3
10 (C)	0.41	0	13.51	0	0.21	0	2
20 (A)	0.29	0	37.06	2	0.17	0	8
20 (C)	0.62	0	13.57	0	0.57	0	10
30 (A)	0.26	0	36.87	2	0.13	0	8
30 (C)	0.6	0	13.03	0	0.45	0	10
40 (A)	0.26	0	48.52	198	0.46	0	12
40 (C)	0.38	0	26.85	0	0.4	0	7

TABLE II

Spacing, NS and Hypervolume Metric Values for Function FDA4

#Particles	**Performance Metric**						**Rank**
	$\overline{S}$	S_{wins}	$\overline{NS}$	NS_{wins}	$\overline{HV}$	HV_{wins}	
2 (A)	0.19	61	14.6	2	$2.84\text{x}10^{12}$	0	4
2 (C)	0.33	2	14.76	2	$1.29\text{x}10^{8}$	0	2
5 (A)	0.19	137	22.39	2	$1.66\text{x}10^{12}$	0	4
5 (C)	0.45	0	21.76	2	$8.29\text{x}10^{10}$	0	4
10 (A)	0.23	0	28.1	2	$3.28\text{x}10^{15}$	0	4
10 (C)	0.41	0	26.93	2	$3.08\text{x}10^{13}$	0	4
20 (A)	0.29	0	33.41	2	$5.82\text{x}10^{17}$	0	12
20 (C)	0.62	0	33.64	2	$2.74\text{x}10^{17}$	0	3
30 (A)	0.26	0	36.32	2	$8.24\text{x}10^{19}$	0	10
30 (C)	0.6	0	36.68	2	$1.58\text{x}10^{18}$	0	4
40 (A)	0.26	0	47.54	200	$5.14\text{x}10^{18}$	2	1
40 (C)	0.38	0	38.34	2	$1.41\text{x}10^{20}$	198	10

TABLE III

Spacing, NS and Hypervolume Metric Values for Function TP1$_{mod}$

#Particles	**Performance Metric**						**Rank**
	$\overline{S}$	S_{wins}	$\overline{NS}$	NS_{wins}	$\overline{HV}$	HV_{wins}	
2 (A)	173.36	0	13.41	2	23.69	0	11
2 (C)	189.33	0	13.24	2	22.6	2	12
5 (A)	117.74	0	15.55	2	25.94	0	7
5 (C)	77.28	0	15.02	2	17.52	0	10
10 (A)	97.55	0	15.41	2	21.62	0	7
10 (C)	73.98	0	15.31	2	17.98	0	7
20 (A)	58.11	0	15.74	2	14.33	0	5
20 (C)	71.38	2	15.38	2	20.33	0	6
30 (A)	56.02	0	15.84	2	13.33	0	2
30 (C)	56.43	4	15.61	2	16.18	0	3
40 (A)	48.78	194	48.52	200	45.22	198	1
40 (C)	70.43	0	16.46	2	18.6	0	4

TABLE IV

Spacing, NS and Hypervolume Metric Values for Function TP2$_{mod}$

#Particles	Performance Metric						Rank
	$\overline{S}$	S_{wins}	$\overline{NS}$	NS_{wins}	$\overline{HV}$	HV_{wins}	
2 (A)	76742.04	1	12.71	2	1.82x10^{13}	2	8
2 (C)	335163.25	0	11.98	2	2.97x10^{14}	198	11
5 (A)	8212.6	0	14.4	2	8.16x10^{10}	0	7
5 (C)	9763.83	2	13.53	2	9.83x10^{11}	0	6
10 (A)	5550.94	3	14.56	2	2.71x10^{10}	0	2
10 (C)	5487.33	0	14.3	2	1.39x10^{10}	0	10
20 (A)	11956.53	0	14.4	2	5.5x10^{10}	0	12
20 (C)	14442.61	0	14.96	2	1.18x10^{11}	0	4
30 (A)	17911.39	0	14.93	2	4.85x10^{11}	0	2
30 (C)	19747.78	0	14.63	2	1.27x10^{11}	0	8
40 (A)	3193.82	194	48.52	200	6.65x10^{10}	0	1
40 (C)	7281.38	0	15.51	2	1.47x10^{10}	0	4

TABLE V

Spacing, NS and Hypervolume Metric Values for various reponses to change for function FDA1

#Particles Re-initialised	Performance Metric						Rank
	$\overline{S}$	S_{wins}	$\overline{NS}$	NS_{wins}	$\overline{HV}$	HV_{wins}	
g_{best}, $g_{secondbest}$ (A)	0.26	0	48.52	200	0.46	2	1
g_{best}, $g_{secondbest}$ (C)	0.38	1	26.85	2	0.4	0	7
10% of population (A)	0.22	0	48.52	200	0.13	0	3
10% of population (C)	0.33	0	28.49	2	0.3	0	5
20% of population (A)	0.27	0	48.52	200	0.21	0	4
20% of population (C)	0.37	0	26.95	2	0.33	0	7
30% of population (A)	0.15	199	48.52	200	0.27	0	2
30% of population (C)	0.39	0	26.73	2	1.05	198	5

TABLE VI

Spacing, NS and Hypervolume Metric Values for various reponses to change for function FDA4

#Particles Re-initialised	Performance Metric						Rank
	$\overline{S}$	S_{wins}	$\overline{NS}$	NS_{wins}	$\overline{HV}$	HV_{wins}	
g_{best}, $g_{secondbest}$ (A)	88022.62	197	47.54	200	5.14x10^{18}	0	2
g_{best}, $g_{secondbest}$ (C)	255519.6	0	38.34	2	1.41x10^{20}	0	7
10% of population (A)	170312.66	0	47.54	200	9.25x10^{19}	0	4
10% of population (C)	226776.26	2	38.31	2	1.4x10^{20}	0	6
20% of population (A)	479245.04	0	47.54	200	1.81x10^{19}	2	3
20% of population (C)	516639.06	1	38.7	2	5.03x10^{21}	1	5
30% of population (A)	479245.04	0	47.54	200	7.49x10^{21}	197	1
30% of population (C)	588408.68	0	36.97	2	2.05x10^{20}	0	8

References

[1] J. Kennedy and R.C. Eberhart. Particle Swarm Optimization. In *Proc. of IEEE International Conference on Neural Networks*, volume IV, pages 1942–1948, 1995.

[2] X. Li and K.H. Dam. Comparing Particle Swarms for Tracking Extrema in Dynamic Environments. In *Proc. of Congress on Evolutionary Computation (CEC 2003)*, volume 3, pages 1772–1779, December 2003.

[3] X. Li, J. Branke, and T. Blackwell. Particle Swarm with Speciation and Adaptation in a Dynamic Environment. In *Proc. of 8th Conference on Genetic and Evolutionary Computation (GECCO 2006)*, pages 51–58, 2006.

[4] T. Blackwell and J. Branke. Multi-swarm Optimization in Dynamic Environments. In *Proc. of Applications of Evolutionary Computing: LNCS volume 3005*, pages 489–500, 2004.

[5] X. Cui et al. Tracking Non-Stationary Optimal Solution by Particle Swarm Optimizer. In *Proc. of the 6th International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing and First ACIS International Workshop on Self-Assembling Wireless Networks (SNPD/SAWN 2005)*, pages 133–138, May 2005.

[6] A. Carlisle and G. Dozier. Adapting Particle Swarm Optimization to Dynamic Environments. In *Proc. of International Conference on Artificial Intelligence (ICAI 2000)*, pages 429–434, 2000.

[7] X. Hu and R.C. Eberhart. Adaptive Particle Swarm Optimization:

TABLE VII

SPACING, NS AND HYPERVOLUME METRIC VALUES FOR VARIOUS REPONSES TO CHANGE FOR FUNCTION $TP1_{mod}$

#Particles Re-initialised	**Performance Metric**						**Rank**
	$\overline{S}$	S_{wins}	$\overline{NS}$	NS_{wins}	$\overline{HV}$	HV_{wins}	
g_{best}, $g_{secondbest}$ (A)	48.78	0	48.52	200	45.22	199	2
g_{best}, $g_{secondbest}$ (C)	70.43	0	16.46	2	18.6	0	6
10% of population (A)	45.71	0	48.52	200	36.49	0	3
10% of population (C)	58.66	0	15.94	2	17.49	0	7
20% of population (A)	34.32	0	48.52	200	36.13	0	3
20% of population (C)	54.73	2	16.3	2	14.99	1	5
30% of population (A)	32.39	196	48.52	200	32.8	0	1
30% of population (C)	78.29	2	15.35	2	17.41	0	8

TABLE VIII

SPACING, NS AND HYPERVOLUME METRIC VALUES FOR VARIOUS REPONSES TO CHANGE FOR FUNCTION $TP2_{mod}$

#Particles Re-initialised	**Performance Metric**						**Rank**
	$\overline{S}$	S_{wins}	$\overline{NS}$	NS_{wins}	$\overline{HV}$	HV_{wins}	
g_{best}, $g_{secondbest}$ (A)	3193.82	0	48.52	200	6.65x10^{10}	0	3
g_{best}, $g_{secondbest}$ (C)	7281.33	0	15.51	2	1.47x10^{10}	1	8
10% of population (A)	6072.46	0	48.52	200	3.93x10^{11}	95	1
10% of population (C)	13270	3	15.93	2	1.11x10^{11}	101	5
20% of population (A)	2872.23	196	48.52	200	4.98x10^{10}	0	2
20% of population (C)	5758.96	0	16.03	2	1.44x10^{10}	0	6
30% of population (A)	2976.68	0	48.52	200	3.75x10^{10}	0	3
30% of population (C)	17990.75	1	15.42	2	1.02x10^{11}	3	7

Detection and Response to Dynamic Systems. In *Proc. of IEEE Congress on Evolutionary Computation (CEC 2002)*, May 2002.

[8] A. Carlisle and G. Dozier. Tracking changing extrema with adaptive particle swarm optimizer. In *Proc. of 5th Biannual World Automation Congress*, volume 13, pages 265–270, 2002.

[9] S-U. Guan, Q. Chen, and W. Mo. Evolving Dynamic Multi-Objective Optimization Problems with Objective Replacement. *Artificial Intelligence Review*, 23(3):267–293, 2005.

[10] J. Mehnen, T. Wagner, and G. Rudolph. Evolutionary Optimization of Dynamic Muli-Objective Test Functions. In *Proc. of 2nd Italian Workshop on Evolutionary Computation and 3rd Italian Workshop on Artificial Life*, 2006.

[11] I. Hatzakis and D. Wallace. Dynamic Multi-Objective Optimization with Evolutionary Algorithms: A Forward Looking Approach. In *Proc. of 8th Annual Conference on Genetic and Evolutionary Computation (GECCO 2006)*, volume 2, pages 1201–1208, July 2006.

[12] X. Li, J. Branke, and M. Kirley. On Performance Metrics and Particle Swarm Methods for Dynamic Multiobjective Optimization Problems. In *Proc. of Congress of Evolutionary Computation (CEC 2007)*, pages 1635–1643, 2007.

[13] M. Cámara, J. Ortega, and J. Toro. Parallel Processing for Multi-objective Optimization in Dynamic Environments. In *Proc. of IEEE International Parallel and Distributed Processing Symposium*, page 243, 2007.

[14] Y. Jin and B. Sendhof. Constructing Dynamic Optimization Test Problems using the Multi-objective Optimization Concept. In *Proc. of Applications of Evolutionary Algorithms: LNCS*, volume 3005, pages 525–536, 2004.

[15] M. Farina, K. Deb, and P. Amato. Dynamic multiobjective optimization problems: test cases, approximations, and applications. *IEEE Transactions on Evolutionary Computation*, 8(5):425–442, October 2004.

[16] E. Zitzler, K. Deb, and L. Thiele. Comparison of Multiobjective Evolutionary Algorithms: Emperical Results. *Evolutionary Computation*, 8(2):173–195, 2000.

[17] K. Deb et al. Scalable multi-objective optimization test problems. In *Proc. of Congress on Evolutionary Computation (CEC 2002)*, volume 1, pages 825–830, 2002.

[18] B. Zheng. A New Dynamic Multi-Objective Optimization Evolutionary Algorithm. In *Proc. of third International Conference on Natural Computation (ICNC 2007)*, volume V, pages 565–570, 2007.

[19] J.D. Schaffer. Multiple Objective Optimization with Vector Evaluated Genetic Algorithms. In *Proc. of the 1st Intenational Conference on Genetic Algorithms*, pages 93–100, 1985.

[20] J.E. Fieldsend and R.M. Everson. Multi-objective Optimisation in the Presence of Uncertainty. In *Proc. of IEEE Congress on Evolutionary Computation*, pages 476–483, 2005.

[21] C.K. Goh and K.C. Tan. An Investigation on Noisy Environments in Evolutionary Multiobjective Optimization. *IEEE Transactions on Evolutionary Computation*, 11(3):354–381, June 2007.

[22] D.A. van Veldhuizen and G.B. Lamont. On Measuring Multiobjective Evolutionary Algorithm Performance. In *Proc. of Congress on Evolutionary Computation (CEC 2000)*, pages 204–211, July 2000.

[23] S-Y. Zheng et al. A Dynamic Multi-Objective Evolutionary Algorithm Based on an Orthogonal Design. In *Proc. of IEEE Congress on Evolutionary Computation (CEC 2006)*, pages 573–580, July 2006.

[24] K.E. Parsopoulos and M.N. Vrahatis. Recent Approaches to Global Optimization Problems through Particle Swarm Optimization. *Natural Computing*, 1(2-3):235–306, 2002.

[25] K.E. Parsopoulos, D.K. Tasoulis, and M.N. Vrahatis. Multiobjective Optimization using Parallel Vector Evaluated Particle Swarm Optimization. In *Proc. of IASTED International Conference on Artificial Intelligence and Applications*, 2004.

[26] M.A. Abido. Two-level of Nondominated Solutions Approach to Multiobjective Particle Swarm Optimization. In *Proc. of 9th Annual Conference on Genetic and Evolutionary Computation*, pages 726–733, 2007.

[27] F .v.d.Bergh. *An analysis of particle swarm optimizers*. PhD thesis, Department of Computer Science, University of Pretoria, 2002.