# Dynamic Multi-objective Optimisation using PSO

Mardé Greeff[1,2] and Andries P. Engelbrecht[2]

[1] CSIR, Meraka Insitute, P.O. Box 395, Pretoria, 0001, South Africa. mgreeff@csir.co.za
[2] University of Pretoria, Faculty of Engineering, Built Environment and Information Technology, Department of Computer Science, Pretoria, 0001, South Africa. engel@cs.up.ac.za

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. Many methods use a weighted average approach to the multiple objectives. However, generally a dynamic multi-objective optimisation problem (DMOOP) does not have a single solution. In many cases the objectives (or goals) are in conflict with one another, where an improvement in one objective leads to a worse solution for at least one of the other objectives. 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. The goal when solving a DMOOP is not to find a single solution, but to find the POF. This chapter introduces the usage of the vector evaluated particle swarm optimiser (VEPSO) to solve DMOOPs. 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 chapter discusses this approach, as well as the effect that various ways of transferring knowledge between the swarms, together with the population size and various response methods to a detected change, have on the performance of the algorithm.

## 1 Introduction

In the world of today optimisation problems occur in a vast variety of situations and aspects of modern life. However, 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. Examples of dynamic optimisation problems can be found in a vast range of scheduling problems, such as timetables, air traffic control, routing in telecommunication networks and target tracking in military operations. Most optimisation problems have more than one objective, e.g. the goals for a manufacturing process might be to maximise the number of products that are manufactured, minimise the time required to manufacture a specific number of products, minimise the time that any machine is idle and minimising the cost. Using a specific machine can be more expensive than another, however the more expensive machine might require less time to manufacture the same number of products than a machine that is cheaper to operate. If you want to manufacture the maximum number of products, using the more expensive machine will minimise the time required, but will increase the cost. Optimisation problems that have more than one objective are called multi-objective optimisation (MOO) problems. Normally these objectives are in conflict with one another, i.e. a better solution for one objective leads to a worse solution for at least one of the other objectives. 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.

When changes occur in the environment that influence the solutions of the MOO problem, the goal becomes to track the changing POF. In order to achieve this, the algorithm should be able to firstly detect that a change has occcurred, and then respond to the change in an appropriate way. If one or more machines used in a manufacturing process (such as the example explained above) break down, it will cause a change in the required solutions and will therefore be an example of a dynamic MOO problem (DMOOP). Comparing one algorithm's performance against another when solving a DMOOP is not a trivial task, since in many real world problems the true POF is unknown. Therefore, benchmark functions with different POF characteristics are used to test the efficiency of an algorithm. To measure the performance of an algorithm, peformance metrics are used.

Greeff and Engelbrecht proposed using the Vector Evaluated Particle Swarm Optimisation (VEPSO) algorithm to solve DMOOPs [1]. When the VEPSO approach is used, each swarm solves only one objective function and then the swarms share knowledge amongst each other. In this chapter the performance of the VEPSO algorithm, using different ways to transfer knowledge between the swarms, are discussed. The effect of the way in which knowledge is transferred between the swarms, in combination with swarm sizes and the effect of various responses to detected changes in the environment, on the performance of the VEPSO algorithm, is also discussed in this chapter.

The rest of the chapter's layout is as follows: Section 2 provides background information and highlights related work that is relevant for the research discussed in this paper. Section 3 provides an overview of the VEPSO algorithm and the changes that has been made to the algorithm for DMOOPs. The experiments that have been conducted are discussed in Sect. 4. The benchmark

functions and performance metrics that have been used to test the algorithm's performance, are highlighted in Sects. 4.1 and 4.2 respectively. The statistical methods that were used for analyses are presented in Sect. 4.3. Section 5 describes the results of the experiments. Finally, Sect. 6 provides a summary and conclusions on the work presented in this chapter.

## 2 Background

Eberhart and Kennedy [2] introduced Particle Swarm Optimisation (PSO), a population-based optimisation method inspired by the social behaviour of bird flocks. Each PSO swarm consists of a number of particles that searches solutions by moving through the search space. Each individual particle has a current position, $\mathbf{x}_i$, velocity, $\mathbf{v}_i$, and personal best position, $\mathbf{y}_i$, where the particle had the smallest error with regards to the objective function. The position amongst all the particles personal best positions that resulted in the smallest error, is called the global best position, denoted as $\hat{\mathbf{y}}$. During each iteration every particle's new position is determined by adding the new velocity to the particle's current position.

Dynamic single-objective optimization problems have successfully been solved using PSO ([3, 4, 5, 6]). When dealing with dynamic problems, where a change in the environment results in a change in the solutions, it is vital that the algorithm can detect that a change has occurred. The concept of a sentry particle has been introduced by Carlisle and Dozier [7]. When using the sentry particle approach to detect whether a change has occurred in the environment, a random number of sentry particles are selected after each iteration. These particles are then re-evaluated before the next iteration, where each particle's current fitness value is compared with its previous fitness value, i.e. its fitness value after the previous iteration. If these two values differ more than a specified value, the swarm is alerted that a change has occurred. Hu and Eberhart [8] suggested that the global best, and global second best particles should be used as sentry particles.

If a change has been detected, the algorithm should respond appropriately to the change. One approach, suggested by Carlise [9], is to re-calculate each particle's personal best. If the new personal best value is less fit than the particles's current position, its personal best value is replaced by its current position. This comparison enables retaining valid past experience. Another approach is to re-initialize a percentage of the swarm's population, preventing that the swarm remains in a small area after the change has occurred, and enabling a portion of the particles to retain their memory, which could be valuable information if the change is small. Cui et al. [6] proposed the usage of an evaporation constant, with a value between zero and one, that is used to update the particle's best fitness value. The particle's memory will gradually decrease over time proportially to the evaporation constant. 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 the evaporation constant approach, the environment is not monitored by any particles, as is the case with the usage of sentry particles.

Much work has been done on using PSO for multi-objective optimisation (MOO) problems. Reyes-Sierra and Coello Coello provides a comprehensive review of the various PSO approaches that were used to solve MOO problems [10]. Recently evolutionary algorithms (EAs) ([11, 12, 13]) and PSO ([14, 15]) have been used to solve DMOOPs. Since changes occur in the environment, the goal is to track the changing POF. In order to test and analyse an algorithm's capability of tracking a dynamic POF, benchmark functions are used. Jin and Sendhof [16] introduced how to define a dynamic two-objective optimisation problem by reformulating a three-objective optimisation test function. Another approach where dynamic problems are created by replacing objectives with new ones at specific times, were presented by Guan et al. [11]. Other benchmark functions were developed by adapting static MOO benchmark functions to dynamic ones. A number of test functions for DMOO based on the static MOO two-objective ZDT functions [17] and the scalable DTLZ functions [18] were developed by Farina et al. [19]. Some adaptions to these test functions were proposed in ([12, 20]). Mehnen et al. [12] proposed DSW functions that are adapted from the static MOO function problem of Schaffer [21] and others added noise to Deb's functions ([22, 23]).

Once the benchmark functions were used to test an algorithm's capability of tracking a dynamic POF, the data has to be analysed and the tracking capability of various algorithms should be compared against one another. 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. The convergence, measured by the generational distance proposed by Van Veldhuizen [24], and spread or distribution of the solutions are often used to measure an algorithm's performance when the POF is known ([23, 25]). The reversed generational distance and the collective mean error were proposed as performance metrics by Branke et al. [14]. Another metric, the $HVR(t)$ metric, represents the ratio of the hypervolume of the solutions and the hypervolume of the known POF at a specific time ([24, 14]). Li et al. [14] proposed a metric of spacing that can be used when the true POF is unknown. 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, was proposed by Cámara et al. [15].

The next section discusses the Vector Evaluated Particle Swarm Optimisation approach that is used to solve MOO problems.

## 3 Vector Evaluated Particle Swarm Optimisation

The Vector Evaluated Particle Swarm Optimisation (VEPSO), a multi-swarm variation of PSO and inspired by the Vector Evaluated Genetic Algorithm (VEGA) [26], was introduced by Parsopoulos et al. [27].

With the VEPSO approach each swarm solves only one objective function and shares its knowledge with the other swarms. The shared knowledge is then used to update the velocity of the particles as indicated in Eqs. (1) and (2) below:

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

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

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; $\hat{\mathbf{y}}^s$ 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; $\mathbf{r_1}, \mathbf{r_2} \in [0,1]^n$; $w^j$ is the inertia weight of the $j$-th swarm; and $s \in [1, \ldots, j-1, j+1, \ldots, m]$ represents the index of a respective swarm. The index $s$ can be set up in various ways, affecting the topology of the swarms in VEPSO.

The VEPSO approach for MOO problems is presented in Algorithm 1. The set of solutions found so far, forming the found POF, are stored in an archive. If the archive is full, and the new solutions are non-dominated and do not already exist in the archive, solutions are selected for removal from the archive to make space for the new solutions based on the distance metric [28].

**Algorithm 1** VEPSO for MOO problems

```
1.  for number of iterations do
2.      peform vepso iteration
3.      if new solutions are non-dominated and don't exist in archive
4.          if space in archive
5.              add new solutions to archive
6.          else
7.              remove solutions from archive
8.              add new solutions to archive
```

In order to solve DMOO problems the algorithm has to be adapted. The changes that were made to the VEPSO approach to solve DMOO problems are discussed next.

### 3.1 VEPSO for Dynamic Multi-objective Optimisation

The VEPSO approach discussed above is used to solve MOO problems. In order to use this approach for DMOO problems, the algorithm must be able

to detect that a change has occurred and then respond to the change in an appropriate manner. Sentry particles are used to detect a change (as discussed in Sect. 2). When a change is detected, a percentage of the swarm's particles are re-initialised. Re-initialisation of a particle entails re-setting its position and then re-evaluating its personal best and the neighbourhood best. The VEPSO approach for DMOO problems is presented in Algorithm 2. The next Section discusses the experiments that were performed to evaluate the ability of this approach to solve DMOO problems.

**Algorithm 2** VEPSO for DMOO problems

```
1.  for number of iterations do
2.      check whether a change has occurred
3.      if change has occurred
4.          respond to change
5.          re-evaluate solutions in archive
6.          remove dominated solutions from archive
7.      perform vepso iteration
8.      if new solutions are non-dominated and don't exist in archive
9.          if space in archive
10.             add new solutions to archive
11.         else
12.             remove solutions from archive
13.             add new solutions to archive
14.     select sentry particles
```

## 4 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 Sects. 4.1 and 4.2 respectively, were used to test the performance of variations of the VEPSO algorithm presented in Sect. 3.1.

To test the effect of various ways of transferring knowledge between the swarms, a ring topology is used, as well as a random selection, i.e. the swarm whose knowledge is used is selected randomly. These different topologies are tested using a varied population size, where the swarms' number of particles are varied between 10, 20, 30 and 40 particles respectively. Furthermore, various responses to change are used, where either 10%, 20% or 30% of the swarm's population is re-initialised. Re-initialisation of particles was either done for all swarms, or only for 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, $\tau_t$, was set to 5 as suggested by Farina et

al. [19], i.e. during each run the functions change every 5 iterations, resulting in 200 changes per run. The PSO parameters were set to values that lead to convergent behaviour [29], namely $w$=0.72 and $c1 = c2 = 1.49$.

### 4.1 Benchmark Functions

To test the performance of VEPSO solving DMOOPs with various types of POFs, with two or more objective functions, four functions proposed by Farina et al. [19] are used. Below, $\tau$ is the generation counter, $\tau_t$ is the number of iterations for which $t$ remains fixed and $n_t$ is the number of distinct steps in $t$.

$$\mathsf{FDA1} = \begin{cases} Minimize : f(\mathbf{x}, t) = (f_1(\mathsf{x}_\mathsf{I}, t), g(\mathbf{x_{II}}, t) \cdot h(\mathbf{x_{III}}, f_i(\mathsf{x}_\mathsf{I}, t), g(\mathbf{x_{II}}, t), t)) \\ f_1(\mathsf{x}_\mathsf{I}) = x_i \\ g(\mathbf{x_{II}}) = 1 + \sum_{x_i \in \mathbf{x_{II}}} (x_i - G(t))^2 \\ h(f_1, g) = 1 - \sqrt{\frac{f_1}{g}} \\ where : \\ G(t) = \sin(0.5\pi t), \;\; t = \frac{1}{n_t} \left\lfloor \frac{\tau}{\tau_t} \right\rfloor \\ \mathsf{x}_\mathsf{I} \in [0, 1]; \;\; \mathbf{x_{II}} = (x_2, \ldots, x_n) \in [-1, 1] \end{cases} \tag{3}$$

The function parameters were set to $n = 20$ and $n_t = 10$ (as suggested by [19]). Function FDA1 (Eq. (3)) has a convex POF where the values in the decision variable space changes, but the values in the objective space remains the same.

$$\mathsf{FDA2} = \begin{cases} Minimize : f(\mathsf{x}, t) = (f_1(\mathbf{x_I}, t), g(\mathbf{x_{II}}, t) \cdot h(\mathbf{x_{III}}, f_i(\mathbf{x_I}, t), g(\mathbf{x_{II}}, t), t)) \\ f_1(\mathbf{x_I}) = x_i \\ g(\mathbf{x_{II}}) = 1 + \sum_{x_i \in \mathbf{x_{II}}} x_i^2 \\ h(f_1, g) = 1 - \frac{f_1}{g}^{(H(t) + \sum_{x_i \in \mathbf{x_{III}}} (x_i - H(t))^2)} \\ where : \\ H(t) = 0.75 + 0.75 \sin(0.5\pi t), t = \frac{1}{n_t} \left\lfloor \frac{\tau}{\tau_t} \right\rfloor \\ \mathbf{x_I} \in [0, 1]; \mathbf{x_{II}}, \mathbf{x_{III}} \in [-1, 1] \end{cases} \tag{4}$$

For FDA2 the parameters were set to the following: $|\mathbf{x_{II}}| = |\mathbf{x_{III}}| = 15$ and $n_t = 10$ (as suggested by [19]). Function FDA2 (Eq. (4)) has a POF that changes from a convex to a non-convex shape, while the values in the decision variable space remains the same.

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

For FDA4 the parameters were set as $n = m + 9$, $|\mathbf{x_{II}}| = 10$, $n_t = 10$ (as suggested by [19]) and $m = 3$ for a 3-objective function. Function FDA4 (Eq. (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.

$$\mathsf{FDA5} = \begin{cases} Minimize : f_1(\mathbf{x}), \ldots, f_m(\mathbf{x}) \\ min_x : f_1(\mathbf{x}) = (1 + g(\mathbf{x_{II}})) \prod_{i=1}^{m-1} \cos(\frac{y_i \pi}{2}) \\ min_x : f_k(\mathbf{x}) = (1 + g(\mathbf{x_{II}}))(\prod_{i=1}^{m-k} \cos(\frac{y_i \pi}{2})) \sin(\frac{y_{m-k+1} \pi}{2}) \\ min_x : f_m(\mathbf{x}) = (1 + g(\mathbf{x_{II}})) \sin(\frac{y_1 \pi}{2}) v \\ where : \\ g(\mathbf{x_{II}}) = G(t) + \sum_{x_i \in \mathbf{x_{II}}} (x_i - G(t))^2 \\ y_i = x_i^{F(t)}, \forall i \in [1, \ldots, (m-1)] \\ G(t) = |sin(0.5\pi t)| \\ F(t) = 1 + 100 \sin^4(0.5\pi t), \ t = \frac{1}{n_t} \left\lfloor \frac{\tau}{\tau_t} \right\rfloor \\ \mathbf{x_{II}} = (x_m, \ldots, x_n); \ x_i \in [0, 1], \forall i = 1, \ldots, n; \ k = 2, \ldots, m-1 \end{cases} \tag{6}$$

Identical function parameters and values as those specified for FDA4 were used here. Function FDA5 (Eq. (6)) has a non-convex shaped POF where the values in both the decision variable space and the objective space changes.

### 4.2 Performance Metrics

This chapter 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 the VEPSO variations are discussed below.

#### Spacing

The metric of spacing [23] indicates how evenly the non-dominated solutions are distributed along the discovered POF, and is defined as

$$S = \frac{1}{n_{PF}} \left[ \frac{1}{n_{PF}} \sum_{i=1}^{n_{PF}} (d_i - \overline{d})^2 \right]^{\frac{1}{2}}, \overline{d} = \frac{1}{n_{PF}} \sum_{i=1}^{n_{PF}} d_i; \quad \overline{S} = \frac{1}{n_r} \sum_{j=1}^{n_r} S_j^i \qquad (7)$$

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

To compare one algorithm against another, the average spacing metric $\overline{S}$ is calculated for each iteration just before a change in the environment occurs.

**Hypervolume**

The S-metric or hypervolume ($HV$) [17] computes the size of the region that is dominated by a set of non-dominated solutions, based on a reference vector that is constructed using the worst objective values of each objective. It is defined as

$$HV(PF) = \cup_{f \in PF} HV(\mathbf{f}) \quad with \quad HV(\mathbf{f}) = \{\mathbf{f}^{'} \in O : \mathbf{f} \prec \mathbf{f}^{'}\} (8)$$

where $PF$ denotes the set of non-dominated sets, $O$ is the objective space and $HV(\mathbf{f})$ is the set of objective vectors dominated by $\mathbf{f}$.

In order to compare one algorithm against another, the $HV$ metric is calculated for each iteration just before a change in the environment occurs. The average over 30 runs is then calculated for each of these iterations.

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

To determine whether there is a significant difference in the performance of one algorithm compared to another algorithm, statistical tests are used as explained in Section 4.3.

### 4.3 Statistical Analysis

To determine whether there is a difference in performance with respect to the performance metrics, a Kruskal-Wallis test was performed for each function. If this test indicates that there is a difference, pairwise Mann-Whitney tests were performed. In all of these tests the average hypervolume value and average spacing value for each of the variations as indicated in Tables 2, 4, 6 and 8 were compaired against the average hypervolume value and average spacing value for each of the variations as indicated in Tables 3, 5, 7 and 9. The $p$-values of these tests can be seen in Table 1.

## 5 Results

This section discusses the results that were obtained from the experiments, with regards to the performance of the variations of VEPSO and the effect of

**Table 1.** $p$-values of Statistical Tests

| Function | Kruskal-Wallis | | Mann-Whitney | |
|---|---|---|---|---|
| | $\overline{S}$ | $\overline{HV}$ | $\overline{S}$ | $\overline{HV}$ |
| FDA1 | 0.73 | 0.6056 | – | – |
| FDA2 | 0.9384 | 0.8519 | – | – |
| FDA4 | **1.237x10$^{-8}$** | **9.54x10$^{-6}$** | **1.101x10$^{-8}$** | **4.93x10$^{-6}$** |
| FDA5 | **0.003097** | **0.0001263** | **0.009095** | **7.868x10$^{-5}$** |

the way in which knowledge is transferred on VEPSO's performance. Furthermore, the effect of the population size and response to a detected change in the environment is highlighted. The results of the experiments can be seen in Tables 2–9 and Figs. 1–4. The overall performance of the knowledge transfer strategies can be seen in Table 10. 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. The values that are printed in bold in all tables indicate the best value for the specific metric. In all figures the filled "$\diamond$" symbol indicates solutions found when 20% of the particles, of only the swarms whose objective function changed, are re-initialised and the "$\times$" symbol indicates solutions found when 20% of the particles of all swarms are re-initialised when a change is detected in the environment (refer to Sect. 5.3).

### 5.1 Overall Performance

Table 10 highlights the overall performance of VEPSO when using either a ring topology or a random topology to exchange knowledge between the swarms. For the 2-objective functions, namely FDA1 and FDA2, there are no real statistical significant differences in VEPSO's performance with regards to the *spacing* and *hypervolume* performance metrics. However, using the random topology for knowledge exchange does lead to a wider spread of solutions for FDA2 as can be seen in Fig. 2. For the 3-objective functions, FDA4 and FDA5, there is a significant difference in performance. This is indicated by the $p$-values in Table 1. The $p$-values that indicate a significant difference in performance are highlighted in bold. From Table 10 it can be seen that the random topology lead to a much higher hypervolume value for FDA4, but a higher spacing metric value. For FDA5, the random topology lead to a lower spacing metric value, but a lower hypervolume value. This indicates that more investigation is required to determine the effect on performance for a wider range of functions. Furthermore, randomly selecting which swarm's knowledge to use (random topology) leads to a wider spread of solutions for FDA4, as can be seen in Fig. 3.

### 5.2 Population Size

From Tables 2–9 it is interesting to note that for the ring topology the best hypervolume values for functions FDA1, FDA4 and FDA5 were obtained when each swarm had 20 particles. For the random topology the highest hypervolume values for functions FDA2, FDA4 and FDA5 were obtained using 40 particles per swarm. When knowledge was exchanged using the ring topology, swarms that consisted of 30 particles lead to the lowest spacing values for functions FDA2 and FDA5. However, for FDA1 and FDA4, the best values for the spacing metric were obtained with 20 and 10 particles respectively. When a random topology was used to exchange knowledge between the swarms, the best spacing values were obtained for functions FDA1, FDA2 and FDA4 using 20 particles per swarm. For FDA5 30 particles per swarm lead to the best spacing values.

### 5.3 Response Strategies

When using a ring topology for knowledge transfer, the best hypervolume values for FDA2, FDA4 and FDA5 were obtained when 30% of the particles were re-initialised of the swarm that is optimising the objective function that has changed. However, when a random topology is used to transfer knowledge between the swarms, the best hypervolume values were obtained when 10% of the particles were re-initialised of the swarm that is optimising the objective function that has changed (refer to Tables 2–9). The best spacing values were obtained for both functions FDA2 and FDA4 re-initialising 20% of the particles of all swarms. For FDA1 and FDA5 the best spacing values were obtained by re-initialising 10% and 30% of the particles of only the swarm whose objective function changed respectively.

**Table 2.** Spacing and Hypervolume Metric Values for Function FDA1

| **%R** | **Ring Topology: #Particles** | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | **10** | | **20** | | **30** | | **40** | | **Avg** | |
| | $\overline{S}$ | $\overline{HV}$ | $\overline{S}$ | $\overline{HV}$ | $\overline{S}$ | $\overline{HV}$ | $\overline{S}$ | $\overline{HV}$ | $\overline{S}$ | $\overline{HV}$ |
| 10A | 0.362 | 6.55 | 0.35 | 9.04 | 0.349 | 8.58 | 0.359 | 10.14 | 0.355 | 8.578 |
| 10C | 0.386 | 8.27 | 0.3 | 8.79 | 0.345 | 8.22 | 0.361 | 9.72 | **0.348** | 8.75 |
| 20A | 0.379 | 8.71 | **0.274** | 6.98 | 0.341 | 8.32 | 0.332 | 8.61 | 0.332 | 8.155 |
| 20C | 0.343 | 10.24 | 0.399 | 10.16 | 0.349 | 8.34 | 0.351 | 8.34 | 0.361 | 9.27 |
| 30A | 0.355 | 8.64 | 0.352 | 9.23 | 0.35 | **10.92** | 0.392 | 9.08 | 0.362 | 9.468 |
| 30C | 0.381 | 10.67 | 0.362 | 10.12 | 0.39 | 10.12 | 0.362 | 8.29 | 0.373 | **9.8** |
| **Avg** | 0.368 | 8.847 | **0.34** | **9.053** | 0.354 | 7.693 | 0.36 | 9.03 | – | – |

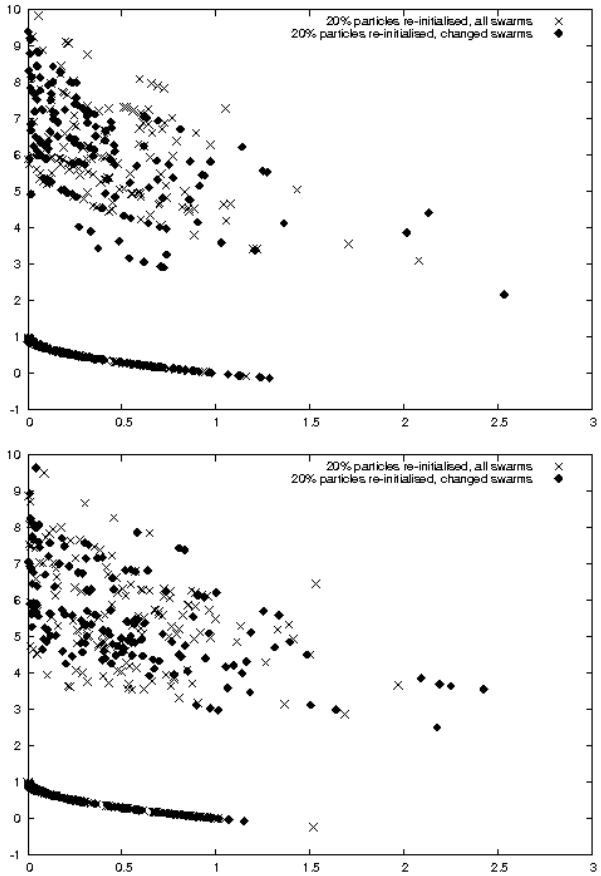


**Fig. 1.** Solutions for Function FDA1 using ring topology on the top and random topology on the bottom

**Table 3.** Spacing and Hypervolume Metric Values for Function FDA1

| **%R** | **Random Topology: #Particles** | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | **10** | | **20** | | **30** | | **40** | | **Avg** | |
| | $\overline{S}$ | $\overline{HV}$ | $\overline{S}$ | $\overline{HV}$ | $\overline{S}$ | $\overline{HV}$ | $\overline{S}$ | $\overline{HV}$ | $\overline{S}$ | $\overline{HV}$ |
| 10A | 0.37 | 9.272 | 0.326 | 9.531 | 0.35 | 7.481 | 0.342 | 8.35 | **0.347** | 8.659 |
| 10C | 0.353 | 7.909 | 0.369 | **10.19** | 0.352 | 7.512 | 0.331 | 9.165 | 0.351 | 8.694 |
| 20A | 0.341 | 7.829 | 0.365 | 9.944 | 0.377 | 8.686 | 0.362 | 9.352 | 0.361 | 8.953 |
| 20C | 0.373 | 9.278 | 0.353 | 8.805 | 0.399 | 8.676 | 0.375 | 10.09 | 0.375 | 9.212 |
| 30A | 0.339 | 8.453 | 0.363 | 8.781 | 0.384 | 8.835 | 0.351 | 7.768 | 0.359 | 8.459 |
| 30C | 0.444 | 9.114 | 0.335 | 9.393 | 0.339 | 9.482 | **0.324** | 8.971 | 0.361 | **9.24** |
| **Avg** | 0.37 | 8.643 | 0.352 | **9.441** | 0.367 | 8.445 | **0.348** | 8.949 | – | – |

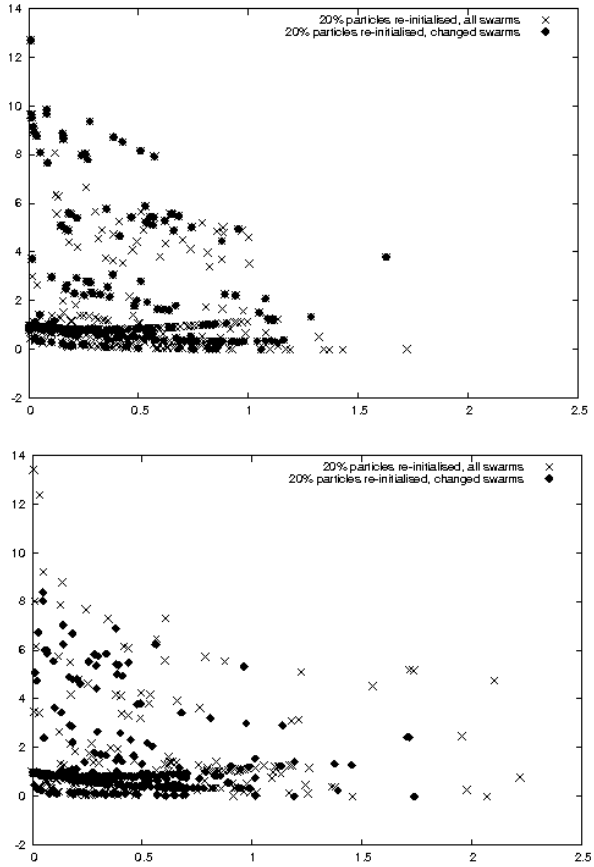


**Fig. 2.** Solutions for Function FDA2 using ring topology on the top and random topology on the bottom

**Table 4.** Spacing and Hypervolume Metric Values for Function FDA2

| **%R** | **Ring Topology: #Particles** | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | **10** | | **20** | | **30** | | **40** | | **Avg** | |
| | $\overline{S}$ | $\overline{HV}$ | $\overline{S}$ | $\overline{HV}$ | $\overline{S}$ | $\overline{HV}$ | $\overline{S}$ | $\overline{HV}$ | $\overline{S}$ | $\overline{HV}$ |
| 10 A | 0.264 | 3.411 | 0.235 | 4.293 | 0.2 | 4.762 | 0.307 | 4.79 | 0.252 | 4.314 |
| 10 C | 0.236 | 5.155 | 0.487 | 9.561 | 0.238 | 4.335 | 0.317 | 5.503 | 0.32 | 4.092 |
| 20 A | 0.33 | 5.957 | **0.146** | 3.034 | 0.269 | 4.574 | 0.223 | 3.106 | **0.161** | 4.17 |
| 20 C | 0.267 | 3.396 | 0.236 | 4.25 | 0.256 | 4.574 | 0.264 | 3.862 | 0.256 | 4.021 |
| 30 A | 0.175 | 2.828 | 0.24 | 4.905 | 0.25 | 3.13 | 0.262 | 3.214 | 0.232 | 3.519 |
| 30 C | 0.207 | 3.958 | 0.243 | 4.399 | 0.248 | **15.995** | 0.253 | 5.74 | 0.238 | **5.015** |
| **Avg** | 0.247 | 4.118 | 0.265 | 5.074 | **0.244** | **6.228** | 0.271 | 4.369 | – | – |

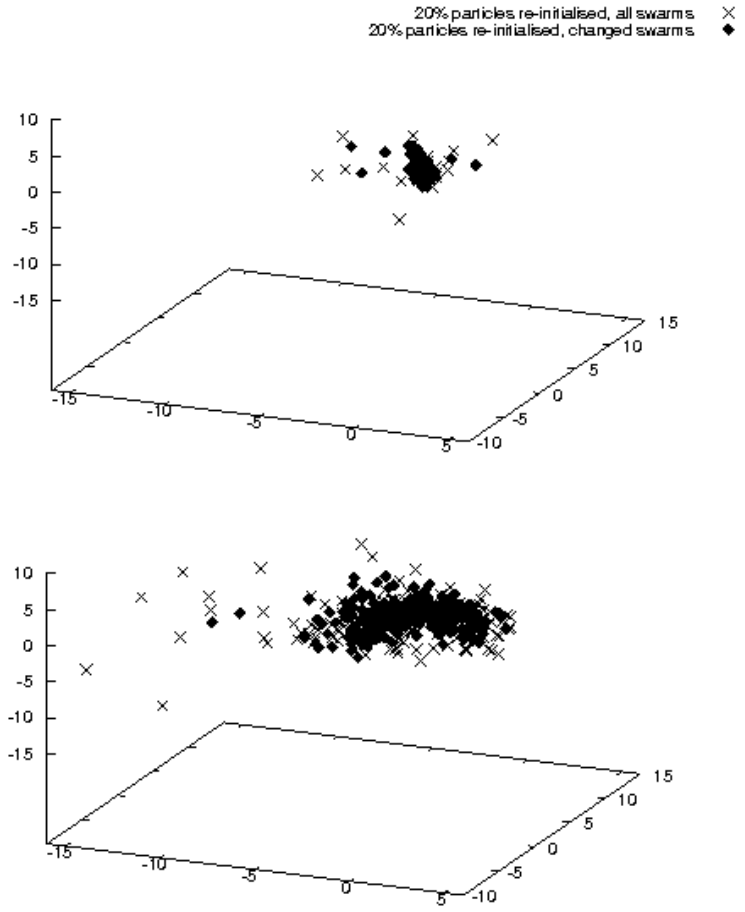


**Fig. 3.** Solutions for Function FDA4 using ring topology on the top and random topology on the bottom

**Table 5.** Spacing and Hypervolume Metric Values for Function FDA2

| **%R** | **Random Topology: #Particles** | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | **10** | | **20** | | **30** | | **40** | | **Avg** | |
| | $\overline{S}$ | $\overline{HV}$ | $\overline{S}$ | $\overline{HV}$ | $\overline{S}$ | $\overline{HV}$ | $\overline{S}$ | $\overline{HV}$ | $\overline{S}$ | $\overline{HV}$ |
| 10 A | 0.251 | 3.963 | 0.253 | 4.513 | 0.219 | 2.357 | 0.284 | **10.77** | 0.252 | 5.401 |
| 10 C | 0.365 | 8.9 | 0.476 | 5.804 | 0.259 | 4.444 | 0.317 | 6.391 | 0.354 | **6.385** |
| 20 A | 0.3 | 4.426 | 0.172 | 3.315 | 0.338 | 6.904 | 0.548 | 7.208 | 0.226 | 5.463 |
| 20 C | 0.247 | 4.028 | 0.193 | 2.489 | 0.263 | 4.71 | 0.215 | 4.881 | **0.153** | 4.027 |
| 30 A | 0.216 | 4.625 | **0.145** | 2.745 | 0.244 | 5.105 | 0.205 | 3.313 | 0.203 | 3.947 |
| 30 C | 0.214 | 3.954 | 0.241 | 3.566 | 0.195 | 2.735 | 0.482 | 5.583 | 0.283 | 3.957 |
| **Avg** | 0.266 | 4.983 | **0.247** | 3.739 | 0.253 | 4.376 | 0.341 | **6.358** | – | – |

**Table 6.** Spacing and Hypervolume Metric Values for Function FDA4

| **%R** | **Ring Topology: #Particles** | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | **10** | | **20** | | **30** | | **40** | | **Avg** | |
| | $\overline{S}$ | $\overline{HV}$ | $\overline{S}$ | $\overline{HV}$ | $\overline{S}$ | $\overline{HV}$ | $\overline{S}$ | $\overline{HV}$ | $\overline{S}$ | $\overline{HV}$ |
| 10 A | 0.19 | 1.08 | 0.19 | 0.39 | 0.19 | 0.24 | 0.03 | 1.19 | 0.15 | 0.72 |
| 10 C | 0.03 | 6.04 | 0.02 | 1.3 | 0.09 | 1230.2 | 0.14 | **5122.9** | 0.07 | 1590.1 |
| 20 A | 0.03 | 9.25 | 0.03 | 0.33 | 0.03 | 1.47 | 0.03 | 0.43 | **0.03** | 2.87 |
| 20 C | 0.04 | 2.60 | 0.03 | 3.22 | 0.03 | 0.71 | 0.03 | 5.62 | 0.03 | 3.04 |
| 30 A | 0.03 | 2.64 | 0.02 | 0.37 | 0.02 | 0.49 | 0.08 | 170.27 | 0.04 | 43.45 |
| 30 C | **0.02** | 0.09 | 0.03 | 9.63 | 0.03 | 1.22 | 0.13 | 4107.2 | 0.05 | **1029.5** |
| **Avg** | 0.06 | 3.62 | **0.03** | 2.54 | 0.06 | 205.72 | 0.07 | **1567.9** | – | – |

**Table 7.** Spacing and Hypervolume Metric Values for Function FDA4

| **%R** | **Random Topology: #Particles** | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | **10** | | **20** | | **30** | | **40** | | **Avg** | |
| | $\overline{S}$ | $\overline{HV}$ | $\overline{S}$ | $\overline{HV}$ | $\overline{S}$ | $\overline{HV}$ | $\overline{S}$ | $\overline{HV}$ | $\overline{S}$ | $\overline{HV}$ |
| 10 A | 0.21 | 249.56 | 0.24 | 274.31 | 0.37 | 4580.01 | 0.27 | 1241.37 | 0.27 | 1586.31 |
| 10 C | 0.46 | **60434.4** | 0.20 | 114.76 | 0.28 | 600.62 | 0.19 | 59.5 | 0.28 | **15302.3** |
| 20 A | 0.21 | 97.98 | 0.20 | 2054.1 | 0.31 | 100.02 | 0.3 | 3080.7 | **0.25** | 1333.2 |
| 20 C | 0.19 | 97.98 | 0.32 | 2054.1 | 0.19 | 100.02 | 0.39 | 3080.70 | 0.27 | 1333.2 |
| 30 A | 0.31 | 1121.8 | 0.17 | 35.95 | 0.27 | 277.75 | 0.31 | 4503.9 | 0.26 | 1484.9 |
| 30 C | 0.33 | 1093.8 | 0.19 | 42.02 | 0.4 | 6379.8 | **0.16** | 25.31 | 0.27 | 1885.2 |
| **Avg** | 0.28 | 10515.9 | **0.22** | 762.52 | 0.3 | **12038.2** | 0.27 | 1998.6 | – | – |

**Table 8.** Spacing and Hypervolume Metric Values for Function FDA5

| **%R** | **Ring Topology: #Particles** | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | **10** | | **20** | | **30** | | **40** | | **Avg** | |
| | $\overline{S}$ | $\overline{HV}$ | $\overline{S}$ | $\overline{HV}$ | $\overline{S}$ | $\overline{HV}$ | $\overline{S}$ | $\overline{HV}$ | $\overline{S}$ | $\overline{HV}$ |
| 10 A | **0.12** | 67.0 | 0.4 | 77.47 | 0.31 | 35.99 | 0.32 | 41.54 | 0.31 | 55.49 |
| 10 C | 0.61 | 52.94 | 0.29 | 201.35 | 0.78 | 1672.76 | 0.68 | 3334.49 | 0.59 | 1315.39 |
| 20 A | 0.55 | 6207.04 | 0.53 | **233672.3** | 0.29 | 20336.39 | 0.63 | 41.75 | 0.5 | **65064.4** |
| 20 C | 0.54 | 30274.45 | 0.48 | 285.33 | 0.41 | 4882.7 | 0.37 | 422.62 | 0.45 | 8966.27 |
| 30 A | 0.47 | 257.3 | 0.49 | 12231.54 | 0.54 | 379.1 | 0.79 | 3844.01 | 0.57 | 4177.99 |
| 30 C | 0.63 | 2185.09 | 0.37 | 249.35 | 0.4 | 478.29 | 0.24 | 452.92 | **0.27** | 841.41 |
| **Avg** | 0.5 | 6507.3 | 0.43 | **41119.55** | **0.32** | 4630.86 | 0.50 | 1356.23 | – | – |

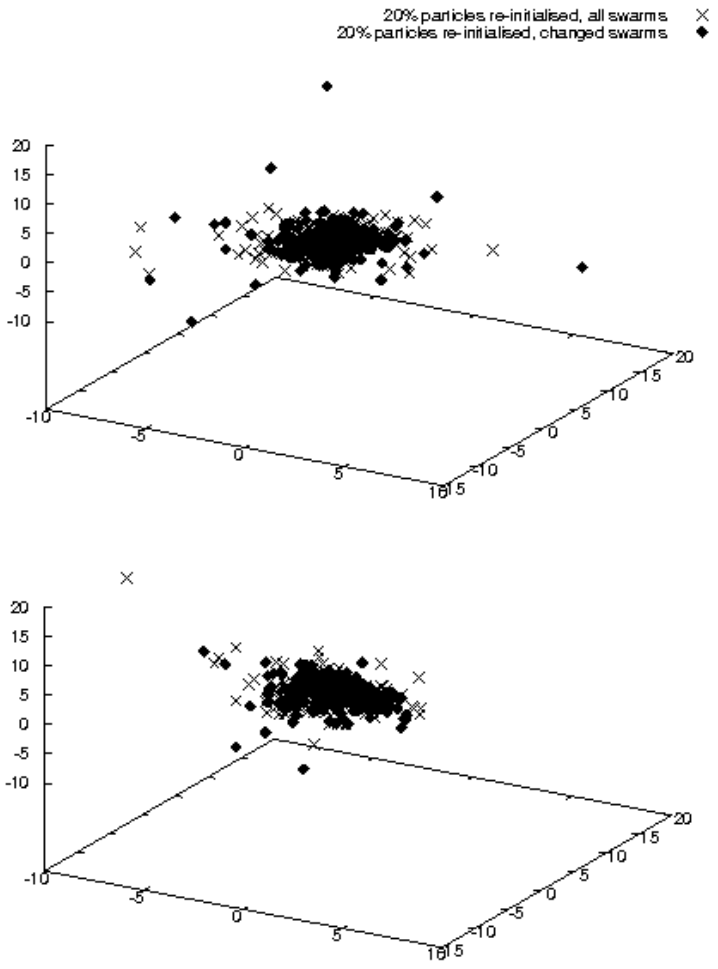


**Fig. 4.** Solutions for Function FDA5 using ring topology on the top and random topology on the bottom

## 6 Summary

DMOO problems occur in a vast range of situations in modern life. The objectives are in conflict with one another and a change in the environment influences the solutions of the optimisation problem. This chapter adapted the VEPSO approach to solve DMOO problems. Two approaches to transfer knowledge between the swarms were presented, namely a ring or random topology. Furthermore, the performance of VEPSO was highlighted using these two knowledge exchange approaches when the number of particles in the swarms, and the response to a change is varied. Results indicated that there

**Table 9.** Spacing and Hypervolume Metric Values for Function FDA5

| %R | Random Topology: #Particles | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | **10** | | **20** | | **30** | | **40** | | **Avg** | |
| | $\overline{S}$ | $\overline{HV}$ | $\overline{S}$ | $\overline{HV}$ | $\overline{S}$ | $\overline{HV}$ | $\overline{S}$ | $\overline{HV}$ | $\overline{S}$ | $\overline{HV}$ |
| 10 A | 0.41 | 53.53 | 0.38 | 48.13 | 0.25 | 23.47 | 0.32 | 22.14 | 0.34 | 36.82 |
| 10 C | 0.37 | 29.51 | 0.3 | 415.95 | 0.33 | 11.97 | 0.94 | **185921.7** | 0.485 | **46594.8** |
| 20 A | 0.41 | 14.71 | 0.62 | 205.28 | **0.16** | 26.92 | 0.27 | 21.95 | 0.37 | 67.21 |
| 20 C | 0.40 | 1092.41 | 0.32 | 110.61 | 0.45 | 275.92 | 0.45 | 169.91 | 0.51 | 412.21 |
| 30 A | 0.28 | 13.11 | 0.19 | 28.81 | 0.16 | 6.83 | 0.32 | 149.66 | **0.24** | 49.61 |
| 30 C | 0.26 | 260.53 | 0.29 | 113.73 | 0.33 | 24.25 | 0.37 | 14.95 | 0.31 | 103.37 |
| **Avg** | 0.35 | 243.97 | 0.35 | 153.75 | **0.28** | 61.56 | 0.45 | **31050.1** | – | – |

**Table 10.** Overall Result Comparison

| **Function** | **Ring Toplogy** | | **Random Topology** | |
|---|---|---|---|---|
| | $\overline{S}$ | $\overline{HV}$ | $\overline{S}$ | $\overline{HV}$ |
| FDA1 | **0.355** | **9.003** | 0.359 | 8.869 |
| FDA2 | **0.256** | **4.947** | 0.277 | 4.864 |
| FDA4 | **0.063** | 444.953 | 0.268 | **3820.845** |
| FDA5 | 0.47 | **13403.485** | **0.357** | 7877.332 |

is not a statistical significant difference in the performance of VEPSO when these two knowledge exchange approaches were used for problems with only 2 objectives. However, when the problem has 3 objectives the random knowledge transfer topology leads to an improvement in either the hypervolume value or the spacing value. Further investigations will be done to determine the effect with a wider range of functions, with a various number of objectives.

Current research focuses on improving the performance of the VEPSO approach for DMOOPs. However, the best version of VEPSO will be compared against other approaches in the future.

## References

1. M. Greeff and A.P. Engelbrecht. Solving Dynamic Multi-objective Problems with Vector Evaluated Particle Swarm Optimisation. In *Proc. of IEEE World Congress on Computational Intelligence (WCCI): IEEE Congress on Evolutionary Computation (CEC)*, pages 2917–2924, June 2008.
2. J. Kennedy and R.C. Eberhart. Particle Swarm Optimization. In *Proc. of IEEE International Conference on Neural Networks*, volume IV, pages 1942–1948, 1995.
3. 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.

4. 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.
5. 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.
6. 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.
7. 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.
8. X. Hu and R.C. Eberhart. Adaptive Particle Swarm Optimization: Detection and Response to Dynamic Systems. In *Proc. of IEEE Congress on Evolutionary Computation (CEC 2002)*, May 2002.
9. 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.
10. M. Reyes-Sierra and C.A. Coello Coello. Multi-Objective Particle Swarm Optimizers: A Survey of the State-of-the-Art. *International Journal of Computational Intelligence Research*, 2(3):287–308, 2006.
11. 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.
12. 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.
13. 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.
14. 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.
15. 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.
16. 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.
17. E. Zitzler, K. Deb, and L. Thiele. Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. *Evolutionary Computation*, 8(2):173–195, 2000.
18. 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.
19. 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.

20. 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.
21. 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.
22. 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.
23. 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.
24. 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.
25. 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.
26. 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.
27. 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.
28. T. Bartz-Beielstein et al. Particle Swarm Optimizers for Pareto Optimization with Enhanced Archiving Techniques. In *Proc. Congress on Evolutionary Computation (CEC)*, pages 1780–1787, 2003.
29. F .v.d.Bergh. *An analysis of particle swarm optimizers*. PhD thesis, Department of Computer Science, University of Pretoria, 2002.