Chapter 7

Dynamic Vector-Based PSO

This chapter investigates the behaviour of the vector-based particle swarm optimization paradigm in changing environments. Multiple optima can change dynamically over time, both temporally and spatially, requiring an algorithm to track these multiple, changing optima. The vector-based PSO, originally developed to locate multiple optima in static environments, is adapted and extended to track multiple optima over time.

7.1 Introduction

PSO can be considered to be a well-established population-based optimization approach. Although PSO has been successfully applied to static problems, real-world optimization often has to be carried out in a dynamic environment; that is, the objective function changes over time. Examples of such problems include scheduling problems such as air traffic control and routing in telecommunication networks. Controlling petrochemical processes also requires frequent re-optimization to balance input and output parameters. Locations of optima and the fitness of the objective function at these positions may change, while some optimal solutions disappear altogether and others appear in new positions. To locate an optimum using particle swarm optimization, particles have to converge on a single point. However, to keep track of dynamically changing optima, some form of redistribution of particles is required in order to increase swarm diversity. This is necessary for the algorithm to continually explore the search space in order to track an optimum of which the position or fitness have changed, or to detect

the disappearance of an optimum or the appearance of a new optimum. Some algorithms such as the charged PSO [8] [9], automatically track optima in a dynamically changing environment, while others require modification of the algorithm.

Depending on the severity of the move, PSO in dynamic environments proved to be effective when locating and tracking a single optimal value [6] [7] [8] [17]. However, in a highly multi-modal environment changes in the fitness of the optima may result in a previous suboptimal solution taking on the best value. Such a situation would be easy to handle if an algorithm could locate and keep track of multiple optima.

This chapter presents an initial and explorative study of the ability of a niching PSO method to track multiple dynamically changing optima. The vector-based PSO [82], [83], [84] is adapted to locate and track optima in a changing environment. A number of scenarios is set up and used to test the performance of the dynamic vector-based PSO in dynamic environments [85].

The chapter is organized as follows: Dynamic optimization problems are defined in section 2, while changes in the environment are discussed in section 3. Research on PSO models to locate a single optimum in a dynamic environment is summarized in section 4. Section 5 presents a discussion of existing algorithms to locate and track multiple dynamic optima. Section 6 describes how the vector-based PSO is extended to locate and track optima in a dynamic environment in parallel, while section 7 concludes the chapter.

7.2 Dynamic optimization problems

A formal definition of a dynamic optimization problem is given as

$$\begin{aligned} &\textit{minimize} \quad f(\mathbf{x}, \varpi(t)), \quad \mathbf{x} = (x_1, \ldots, x_{n_x}), \qquad \varpi(t) = (\varpi_1(t), \ldots, \varpi_{n_w}) \\ &\textit{subject to} \qquad g_m(\mathbf{x}) \leq 0, m = 1, \ldots, n_g \\ &\qquad\qquad\qquad h_m(\mathbf{x}) = 0, m = n_g + 1, \ldots, n_g + n_h \\ &\qquad\qquad\qquad x_j \in dom(x_j) \end{aligned} \tag{7.1}$$

where $\varpi(t)$ is a vector of time-dependent objective function control parameters. The objective is to find

$$\mathbf{x}^*(t) = \min_x f(\mathbf{x}, \varpi(t)) \tag{7.2}$$

where $\mathbf{x}^*(t)$ is the optimum found at time step t.

Dynamic systems change over time in several ways. Changes can take place at regular or irregular time intervals, or continuously. The timescale involved is referred to as *temporal*

severity. Changes in optimal positions can also be quantified. The term *spatial severity* is used to indicate the amount of change in the position of an optimum at each step. The severity parameter, ζ, is used to indicate the change in location before the next optimization effort.

In addition to the type of dynamic environment, the way in which change occurs, can also vary. The next section presents a discussion of changes in the environment.

7.3 Changes in the environment

To a large extent the dynamics of an environment depends on the optimization problem. Optimization algorithms for dynamic environments are designed to react to changes in the environments. If changes occur at specific intervals, the mechanism the algorithm uses to improve exploration is invoked to coincide with those times. If changes do not occur at fixed intervals, those changes in the environment need to be detected automatically.

Carlisle and Dozier [17] [18] used a sentry particle to detect change in the environment. A new sentry particle is chosen at random each iteration. If the fitness of such a particle is different from that of the previous iteration, the environment has changed, although no change in the fitness of the sentry particle does not guarantee that the environment did not change. If the change is significant, a tracking strategy is triggered. An alternative method, proposed by Hu and Eberhart [45], monitors the global best position. The method was improved by monitoring the second-best position as well [46].

Response to changes in the environment can take many forms. If the environment changes while the swarm is still in the process of converging, and the spatial severity is not too large, the PSO automatically adapts and moves towards the new optimum [33]. If the swarm has converged to an equilibrium state, which is possible if the temporal severity is low, the swarm becomes stable and will not move to the new optimal position. In such cases some strategy has to be deployed to inject more diversity which will allow the swarm to move to the new optimal position. According to Blackwell [6], optimization with particle swarms has two major ingredients, the particle dynamics and the particle information network. A combination of these ingredients make PSO a robust and efficient optimizer of real-valued objective functions. However, when applying PSO to dynamic problems the algorithm must be modified for optimal results, as *diversity loss* is experienced due to convergence, while the problem of *outdated memory* occurs as a result of the environment dynamism. Diversity loss is the more serious problem, as the swarm has to re-diversify, locate the shifted optimum, and re-converge. Two strategies

can be utilized to address such a problem, namely the deployment of a re-diversification mechanism when the objective function changes, and/or some diversity measure that is maintained during the run.

The next section overviews a number of approaches to single-solution particle swarm optimization in dynamic environments.

7.4 Particle swarm models for tracking a single solution in a dynamic environment

Carlisle and Dozier [17] [18] conducted one of the first investigations into modification of the Particle Swarm Optimizer to locate changing extrema. Each particle's record of its personal best position is reset to the current particle position as the environment changes. Resetting only takes place if, after the environment change, the personal best position is worse than the current position. Thus direction and velocity decisions based on outdated information are avoided. This process is initiated by two methods: periodic resetting, based on the iteration count, and triggered resetting, based on the magnitude of the change in the environment. For the second method a sentry particle is selected at random in the search space. At each iteration the sentry particle's fitness is reevaluated; if the fitness changed by more than the *trigger value*, the personal best positions of all particles are reset.

It must be noted that such a strategy will only be effective when the swarm has not yet converged to a solution. The use of a sentry particle to detect change is also not reliable, as localized fluctuations may occur.

Eberhart and Shi experimented with tracking and optimizing a single optimum in a dynamic system [30]. Successful tracking of a 10-dimensional parabolic function is demonstrated. PSO was perceived to perform better than genetic algorithms on similar problems. However, dynamic environments can vary considerably, and many real-world systems change state frequently, thus requiring frequent re-optimization. Eberhart and Shi reason that searching from the current position works well for small environmental changes, or when the swarm has not yet reached an equilibrium state. For severe environmental changes, re-initialization of the entire swarm can be more effective as previous optimal positions will contribute very little or nothing at all to the effort. A combination of these two approaches, namely retaining the global best position and re-initializing a percentage of the particle positions, is suggested in order to retain

potentially good positions as well as increase diversity.

Blackwell and Branke identify the following problems which have to be considered when applying PSO to dynamic environments [9]:

Outdated memory: When the objective function changes, an individual particle's personal best solution (called the *particle attractor* by Blackwell and Branke) may no longer apply, and may even guide the search in the wrong direction. This problem is usually solved by resetting the position of the attractor to the current particle position, as described earlier.

Diversity loss and linear collapse: One of the principle characteristics of PSO is convergence to a stable point as proved by Van den Bergh and Engelbrecht [100], Clerc and Kennedy [20], and Trelea [97]. However, as pointed out by Van den Bergh [100], this point is not necessarily an optimum. A shrinking swarm is synonymous with loss of diversity. If the optimum moves away, but the new position is still within the collapsing swarm, there is a good chance that the swarm will successfully track the moving target [6]. If the new target is significantly far from the collapsing swarm, the low velocities of the particles will inhibit re-diversification and tracking. Van den Bergh and Engelbrecht [100], and Blackwell and Bentley [8] have found that the swarm can even oscillate around a false attractor and along a line perpendicular to the true optimum, a feature known as linear collapse.

Blackwell and Branke categorize approaches to counterbalance the effect of diversity loss as follows [9]:

Introduce diversity after the problem has changed: The entire, or part of the swarm is randomized after each function change, or at some predetermined time interval. Such an approach might have the effect that useful information such as positions with good fitness is lost.

Maintain diversity throughout the run: Blackwell and Bentley [8] have introduced the charged PSO, an attempt to maintain diversity throughout the run. A nucleus of neutral particles act like a conventional PSO exploring the neighbourhood to locate an optimal solution. In addition, a roaming swarm of "charged" particles, that is, particles that are repelled by a subswarm converging on a peak, is maintained to detect new peaks in the search space. In a later innovation, Blackwell and Branke [10] introduced the

idea of quantum particles that move to random positions around the swarm's global best position.

Maintain multiple swarms on different peaks: Blackwell and Branke [9] constructed interacting multi-swarms by extending the single population PSO as well as the charged particle swarm optimization method. A number of subswarms are maintained around local optima that have been located, while a swarm of charged particles search for new local optima, where, if discovered, new subswarms are established. Functions where the landscapes consist of several peaks and changes to the locations, heights and widths of the peaks are small, are expected to respond well to such a strategy. If peak heights change in such a way that a suboptimal solution becomes the solution with the best fitness in the search space, it is not difficult to keep track of the global optimum. A further refinement entails sustaining diversity within subswarms by incorporating the strategy used by quantum particles.

Blackwell and Branke extended the idea of multi-swarms by proposing two forms of swarm interaction, namely *exclusion* and *anti-convergence* [7]. Exclusion prevents swarms from settling on the same peak by re-initializing the lesser swarm if two swarms move too close to one another. Anti-conversion re-initializes the worst swarm in order to track down any new peak that may appear.

Although the purpose of the multiswarms that are maintained on different peaks is to track the overall optimal solution in a dynamic environment, the creation of sub-swarms *per se* is relevant to multimodal optimization. Tracking multiple dynamic optima in parallel exhibit the same advantages as the multi-swarm approach, but starts from another premise.

The next section describes how a multimodal particle swarm optimizer can be adapted for dynamic environments.

7.5 Tracking multiple optima in a dynamic environment

In multidimensional systems changes can occur in one or more dimensions, independently or simultaneously. Changes occurring in a problem space with many suboptimal solutions can also have the effect that a suboptimal solution becomes the optimal solution and vice versa. Thus, even if the severity is small, the location of the optimum may change considerably. In most cases this problem will not occur if a niching algorithm is adapted for dynamic systems,

where the objective is to keep track of all optima. The best solution can easily be selected from all the tracked optima at any stage during the optimization process. A possible exception can occur if a new optimum appears and immediately becomes the optimal solution. However, once the new optimum is detected and added to the collection of optima, the process will proceed normally.

7.5.1 A dynamic test function generator

Dynamic environments can be generated by the moving peaks benchmark (MPB). De Jong and Morrison's test problem generator [25] was devised to be used in the study of evolutionary algorithm performance in changing environments. The problem generator is also suitable to evaluate PSO performance in dynamic environments, especially if the problem space contains a number of suboptimal solutions as well. An environment in two dimensions consisting of a number of cone shapes is generated by the following equation:

$$f(x_1, x_2) = \max_{i=1,\ldots,N} \left(H_i - R_i \sqrt{(x_1 - x_{1_i})^2 + (x_2 - x_{2_i})^2} \right) \tag{7.3}$$

where the height of each cone is given by H_i, the slope by R_i and the position by (x_{1_i}, x_{2_i}). The number of optima, their positions, shapes and heights can be specified. An environment with three cones is illustrated in Figure 7.1.

7.5.2 PSO models for multiple dynamic optima

Particle swarm optimizers that locate multiple optima in parallel provide natural mechanisms for adapting to dynamic environments. The species-based PSO [57] described in Chapter 4 is eminently suitable for such a modification.

Parrott and Li [70] adapted the algorithm to track multiple peaks simultaneously. The resulting algorithm, referred to as the dynamic species-based PSO, is based on the static species-based PSO where the particle with the best fitness is identified as a species seed and a subpopulation is formed with particles within a set speciation radius, r. The process is repeated until no more particles remain. The result is a number of subpopulations, each with its own species seed which is the particle with the best fitness in the subpopulation. At each iteration of the algorithm, particles are updated and subpopulations reformed. If the function changes, species seeds will be identified at new positions, thus providing a natural way in which the positions of peaks in the landscape can be tracked. To allow particles to move towards the new

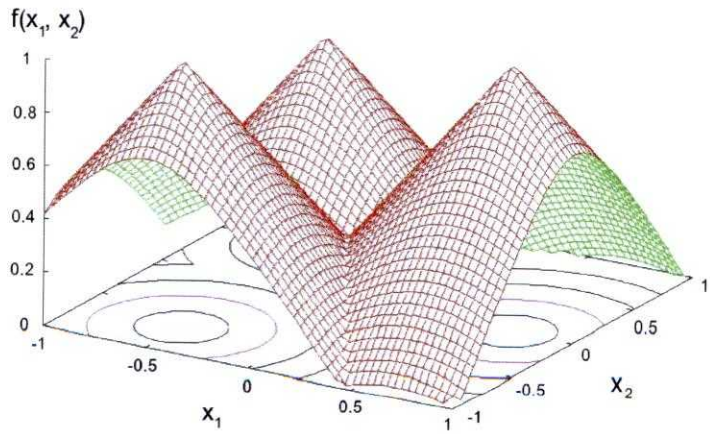

Figure 7.1: An environment with three peaks at positions $(-0.5, -0.7)$, $(-0.5, 0.5)$ and $(0.5, 0)$, generated by Morrison and De Jong's test problem generator

species seeds, the dynamic SPSO re-evaluates each particle's personal best fitness value before updating the positions of the particles. Crowding at known optima had to be prevented so that some particles could be allocated to search for new optima. Such a strategy entails introducing a maximum species population parameter. Only the best candidate members are allocated as members of a species, while redundant particles are re-initialized at random positions in the solution space.

The dynamic SPSO was extensively tested on functions generated by De Jong and Morrison's dynamic test function generator [25] [70]. Several parameters were varied and results reported. These results indicate that the algorithm can successfully track optima in a dynamic two-dimensional environment.

Li *et al.* proposed an improved dynamic species-based PSO that incorporates techniques to improve its tracking ability [59]. Some of these techniques were suggested by the multi-swarm approach of Blackwell and Branke discussed in section 6.4. Inspired by the quantum swarm model, some particles, the so-called "quantum" particles, are positioned around each peak to maintain diversity. An *anti-convergence* method to re-randomize the worst swarm when all species have converged is also adopted by the species-based PSO. The method is invoked if the average particle diversity falls below a certain threshold. Different regions of the search space can then be explored to search for new peaks. The SPSO was tested with a variety of scenarios set up by the moving peak benchmark [25]. The results suggested that SPSO can adapt well

on most test cases.

7.6 The dynamic vector-based particle swarm optimizer

This section describes how the vector-based PSO (referred to as the VBPSO) is extended to locate and track optima in a dynamic environment in parallel.

The proposed dynamic vector-based PSO uses the VBPSO to find initial multiple optima. These optima have to be tracked when the objective function changes, causing optima to move to other positions. For a tracking strategy to be efficient, all optima should be found after each objective function change with less effort than re-optimization would entail. The dynamic VBPSO endeavours to retain useful information such as previous positions of optima after each modification of the objective function, thus avoiding complete re-optimization. If changes are not too severe, the positions where the previous optima were located, are retained and act as starting points to track optima that have moved away. Thus fewer particles are required, making the process more efficient. For severe changes it could be argued that no benefit can be derived from previous optimal positions and that complete re-optimization would be preferable.

Each time the objective function is modified and the landscape changes, a tracking mechanism is invoked. In the dynamic vector-based PSO this mechanism consists of two stages. Stage 1 tracks existing optima that have moved away, or of which the fitness at one or more of the peaks have been modified. Provided the severity is not too large, optima can be tracked with relatively little effort. Stage 2 of the algorithm contains a technique to locate new optima that may have appeared.

The process to locate and track optima in a dynamic environment is described below while algorithm 12 formally presents the vector-based PSO algorithm for multiple dynamic optima.

Locate niches and optimize: After the swarm had been initialized by creating an appropriate number of particles, the static vector-based PSO as described in chapter 5, is used to locate and demarcate niches sequentially and optimize these niches in parallel. A small problem-dependent parameter, the *granularity*, described in section 5.5.2, is set in advance to facilitate niching. Thus the initial positions and fitnesses of all the optima are obtained. Faster changes in the environment may result in less accurate optimal positions.

Track existing optima: In a dynamic environment optima are tracked by finding new optimal positions each time a modification of the objective function is detected, or at specific

Algorithm 12 The dynamic vector-based PSO

```
begin
    Initialize the swarm by creating n particles;
    Locate niches and optimize using the vector-based PSO;
    while function is changing do
        if function has been modified then
            Stage 1:
            Discard all particle information except previous optimal positions;
            Create particles at previous optimal positions;
              Establish new personal best positions (pbest);
              Set each neighbourhood best position (nbest) to corresponding pbest;
              Update position vectors towards pbest, nbest and and their dot
              products;
            Create 3 additional particles near each optimal position to form small
            subswarms;
            Optimize subswarms in parallel to converge on new positions;
            if a peak has disappeared then
                Merge niches;
            end
            Stage 2:
            Discard all particle information except optimal positions found in stage 1;
            Create n particles over entire search space;
            for all existing niches do
                Find niche radii;
                Mark particles within niche radii as belonging to the corresponding
                niche;
            end
            for particles not assigned to a niche do
                repeat
                    Find particle with best pbest;
                    Set nbest of all particles to best pbest;
                    Find niche radius;
                    Mark particles within niche radius as belonging to the corresponding
                    niche;
                until all particles have been included in a niche;
                Optimize niches in parallel and merge if necessary;
            end
        end
    end
end
```

intervals if the function changes continuously. At this stage a number of particles will have converged on each previous optimal position. Redistribution of these particles throughout the search space is instrumental in tracking optima when the environment changes. However, if the locations of the optima do not change too much, only a few particles in the vicinity of each existing optimum are required to track the optima to their new positions.

The dynamic vector-based PSO employs a strategy where only the existing neighbourhood best position of each niche is retained when the objective function is modified and optima need to be tracked. All other particle information is discarded. After the objective function has changed, a new particle is created at each previous neighbourhood best position. Similar to the procedure the static vector-based PSO employs to create initial particles, a personal best position is associated with the particle at each previous neighbourhood position. A random position is found near to each particle and the personal best is set to the fittest of the two positions while the less fit position becomes the particle position. The neighbourhood best position associated with each of these particles is set equal to the personal best position of that particle. Such steps are necessary since the personal best position of each previous optimum has to be re-evaluated in a changed environment. At this stage one particle has been created at each previous neighbourhood best position so that the number of particles is equal to the previous number of optima.

To enable the algorithm to track an optimum, a small number of additional particles are created in the immediate vicinity of each particle created at a previous optimal position. These particles are initialized at random positions within a radius r from the neighbourhood best particle. The value of r is equal to the *granularity* that has been set in advance for the objective function that is being optimized. Thus a small subswarm is formed in each niche. Appendix A.3 presents empirical results showing that a subswarm consisting of the original neighbourhood best particle as well as three additional particles, giving a total of four per niche, is able to effectively track a moving optimum. New personal best positions are updated. The neighbourhood best position of each additional particle is set to the neighbourhood best position of the first particle in each niche. Vectors pointing to the personal best and neighbourhood best positions, as well as their dot product, are calculated.

To locate modified optimal positions, the subswarms thus created, are optimized in parallel. During the optimization process subswarms naturally move towards better positions.

It may happen that an optimum moves into a neighbouring niche and disappears. Re-evaluation of the personal best positions of the subswarm associated with the peak that disappeared causes that subswarm to move towards the neighbouring optimum where the two subswarms will eventually merge.

The strategy used to track existing optima is referred to as stage 1 of the algorithm in the formal description of the algorithm. New optima appearing in the landscape during modification of the objective function, are not discovered during stage 1.

Locate new optima: New optima may appear when the objective function changes. Locating these optima is not a simple matter, as the initial strategy to find niches must basically be repeated to find new candidate solutions. The dynamic VBPSO algorithm includes a mechanism to locate new optima. This mechanism is invoked after stage 1 of the algorithm and is referred to as stage 2. These stages are invoked each time the objective function is modified or at specific intervals in the process. Stage 2 of the algorithm only retains the positions of the optima located during stage 1. All other particles and their associated information are discarded. Similar to the static VBPSO that was used to locate initial optima for one run of the algorithm, new particles are created at random positions throughout the search space. The number of particles is set equal to the original number created in the first phase of the algorithm. Niches are established by using the vector dot product to calculate niche radii. However, instead of repeating the entire sequential process to find candidate solutions, niche radii of the existing optima are first calculated. Particles falling within those radii, are marked as belonging to that niche. If a new peak has appeared near to an existing optimum, the niche radius of the existing optimum will be smaller and not include particles expected to converge on the new peak. To reduce the effort of finding new optima, the subswarms thus formed, are not optimized again. The sequential process to find candidate solutions and calculate their niche radii is then resumed for the remaining particles. Only subswarms formed by these remaining particles are optimized during the optimization phase. If new optima have appeared, subswarms would have formed around those optima on which it will converge during optimization. However, if no new optima have appeared after the last modification of the objective function, particles outside the niches surrounding the existing optima will form subswarms adjacent to those niches. These subswarms reside in false niches that exhibit a tendency to move into existing niches, and will eventually converge on existing optima.

Clearly, the process to locate new optima is computationally more expensive than the

process to track existing optima. The question arises whether complete re-optimization at set intervals might not be considered as a workable alternative. The next section investigates these possibilities.

7.7 Conclusion

This chapter presented a technique to extend the vector-based particle swarm optimizer to be applicable to dynamic environments. Particle swarm models for tracking a single solution in a dynamic environment were discussed while an overview of PSO models for multiple dynamic optima was presented. The vector-based PSO for multiple dynamic optima was described and formally presented as the dynamic vector-based PSO algorithm. Demonstration of its ability to track multiple moving optima in a number of selected dynamic environments is presented in chapter 8.