Chapter 11
Improving Local Convergence in Particle Swarms by Fitness Approximation Using Regression

Stefan Bird and Xiaodong Li*

Abstract. In this chapter we present a technique that helps Particle Swarm Optimisers (PSOs) locate an optimum more quickly, through fitness approximation using regression. A least-squares regression is used to estimate the shape of the local fitness landscape. From this shape, the expected location of the peak is calculated and the information given to the PSO. By guiding the PSO to the optimum, the local convergence speed can be vastly improved. We demonstrate the effectiveness of using regression on several static multimodal test functions as well as dynamic multimodal test scenarios (Moving Peaks). This chapter also extends the Moving Peaks test suite by enhancing the standard conic peak function to allow the creation of asymmetrical and additional multiple local peaks. The combination of this technique and a speciation-based PSO compares favourably to another multi-swarm PSO algorithm that has proven to be working well on the Moving peaks test functions.

Keywords: Particle Swarm Optimization, Swarm Intelligence, Optimization in Dynamic Environments, Regression Techniques, Numerical Optimization.

11.1 Introduction

Local search methods are known for their extremely fast convergence, however they are also highly susceptible to becoming trapped in the first optimum they find. Many real world problems are far too complex to be solvable by these methods. Evolutionary Algorithms (EAs) and Particle Swarm Optimisation (PSO) have been proved

Stefan Bird
School of Computer Science and IT, RMIT University, Melbourne, Australia
e-mail: stbird@seatiger.org

Xiaodong Li
School of Computer Science and IT, RMIT University, Melbourne, Australia
e-mail: xiaodong.li@rmit.edu.au

* Corresponding author.

Y. Tenne and C.-K. Goh (Eds.): Computational Intel. in Expensive Opti. Prob., ALO 2, pp. 265–293.
springerlink.com

to be effective search strategies. Both algorithms are able to converge on an optimum even when the catchment area occupies only a small portion of the search space. These algorithms are far less likely to become trapped in a local peak than local search, especially when combined with a diversification measure. This property comes at a cost though, as they take far more evaluations to locate an optimum than local search methods.

Combining local search with an EA (or PSO) can provide the best of both worlds, as we gain the robustness of the population-based algorithms as well as the local search's convergence speed [26]. This hybrid approach is quite common, for example [24, 34, 36]. However, using the local search requires extra fitness evaluations to be performed; when considered over the entire optimisation process, these evaluations can be very costly.

To overcome the issue associated with high computational cost, several fitness approximation techniques have been developed (see also Section 11.2.1). For example, in aerodynamic structure optimization, since simulations for computational fluid dynamics are usually very expensive, approximate models were developed [1, 16]. Fitness approximation was also used in conjunction with an evolutionary algorithm for protein structure prediction to cut down the computational cost [27]. Another interesting study in [23] shows the benefits of approximating fitness landscape using a polynomial regression model.

This chapter presents a fitness approximation technique that helps Particle Swarm Optimisers (PSOs) locate an optimum more quickly, without requiring any extra fitness evaluations. Rather than performing a local search, we use the candidate solutions already tested by a PSO to create a surface that best fits the peak. We then attempt to calculate the highest point of this surface. Provided that the local features of the fitness landscape roughly match our surface, the optimum should be very close to the computed highest point. This allows us to very quickly hone in on the actual maximum point – with each successive attempt we know more and more about the landscape, improving our estimation further.

Our early study in [7] suggested that regression was able to improve efficiency in handling some dynamic optimization functions (ie., Moving peaks scenario 2). Extending the early findings, this chapter provides several new investigations on using regression. Firstly, we identified similarities and differences between our regression method and other existing works in literature. Secondly, we included several widely used static multimodal test functions to further verify if regression is effective in improving local convergence for solving static multimodal problems in general. In addition, we also adopted a Generic Hump function (which is tunable with the number of peaks and the number of dimensions) to study especially if regression is effective in reducing the number of evaluations for high dimensional multimodal problems. Furthermore, we extended the Moving Peaks test functions to allow creation of asymmetrical and additional multiple local peaks. The effectiveness of regression on these more complex peak shapes was examined.

Although fitness approximation using regression has been developed for EAs [23], this study represents a first attempt to integrate regression with a PSO for improving local convergence. This paper is organised as follows. Section 11.2 provides

the background of this technique and the algorithms we have used to test it. A detailed explanation of the method follows in Section 11.3. Sections 11.4 and 11.5 show the experimental setup and results. Our conclusions will be presented in Section 11.6, as well as some further research directions.

11.2 Background

Local search methods are typically designed to rapidly locate an optimum once its general area has been found. These methods are susceptible to becoming trapped in a local optimum, meaning that they are most effectively used once the peak's location is already approximately known. The most intuitive local search method is hill climbing. This method works by continually sampling the decision space around the best point found so far [26]. At each iteration, a point somewhere near the current best is selected and evaluated. If the new point is better than the current best it replaces it, otherwise the new point is discarded. By repeating this process many times we "climb" the peak of the initial starting point, usually chosen randomly. One variant of hill climbing is gradient ascent, which works by using the derivative of the fitness function to guide the search direction [14]. The next point chosen to search is one that is close to the last point evaluated, but in the direction of the steepest ascent. However, this technique can only be used with fitness functions where the gradient can be computed.

To improve the local convergence of a PSO on a multimodal fitness landscape we are incorporating a method that approximates the fitness landscape using regression. This chapter will demonstrate that the use of regression and a convergence enhancer can dramatically improve local convergence while saving computational cost. This section provides the background for these techniques.

11.2.1 Fitness Approximation

In recent years, several studies have proposed EAs incorporating fitness approximation with an aim to improve performance while not incurring expensive computational cost [15, 23, 28]. Typically these EAs employ a *surrogate* model in place of the expensive original function evaluations. The surrogate model is used to approximate the original fitness function by using a small set of evaluated search points chosen from the EA population. The goal is to reduce the number of expensive original function evaluations while retaining an accurate approximation of the original function. The most commonly used techniques for constructing surrogate models include Kriging [31], neural networks [17], and polynomial regression [23, 32, 37]. A recent survey on fitness approximation in EAs can be found in [15].

One particularly interesting EA combining fitness approximation and local search is EANA (Evolutionary Algorithms with N-dimensional Approximation) [23], where polynomial regression was used to approximate fitness landscape. The experiments

of EANA on a wide range of test functions have demonstrated the effectiveness of this approach. Nevertheless, EANA was designed to locate a single global optimum (not multiple global optima), and test functions were all static test functions. To the best of our knowledge, no EAs using fitness approximation and local search have been tested for locating multiple global optima. It is even more difficult when tracking multiple moving peaks in a dynamic environment. This paper aims to develop a PSO incorporating fitness approximation and local search methods, and evaluate the effectiveness of the hybrid PSO using multimodal test functions in both static and dynamic environments.

11.2.2 Particle Swarms

Particle Swarm Optimisation (PSO) is an evolutionary algorithm that mimics a flock of birds [18]. As birds move throughout a territory, they are all simultaneously watching for both food and predators. In addition they are monitoring the behaviour of the birds around them. A change in a neighbour's behaviour usually indicates there is some new information available, for example a food source or predator has been seen. By copying the behaviour of its neighbours each bird is able to benefit from the discovery, even before it has the information itself. In PSO, each bird is represented by a particle. It maintains its current location and velocity, as well as a memory of the best location it has seen so far, known as the *personal best*. Each particle also has a number of neighbours with whom it can share its personal best. At every timestep each particle chooses a random point between its personal best location and the fittest personal best of any of its neighbours. It then steers towards that point, but does not travel there directly. The particles have momentum, meaning that if the chosen point is in the opposite direction to where they're travelling it may take a number of timesteps to turn around. This ensures the particles thoroughly explore the area surrounding the fittest known point, and are able to jump to a better peak if one is discovered nearby.

To guarantee convergence, we have used Clerc's constriction coefficient PSO [11, 18]. This is described by Equations (11.1) and (11.2), which are run for every timestep t.

$$v_{(i,j,t+1)} = \chi(v_{(i,j,t)} + \varphi_1(p_{(i,j,t)} - x_{(i,j,t)}) + \varphi_2(p_{(g,j,t)} - x_{(i,j,t)})) \tag{11.1}$$

$$x_{(i,j,t+1)} = x_{(i,j,t)} + v_{(i,j,t+1)} \tag{11.2}$$

where:

$$\varphi_1 = c_1 r_1, \quad \varphi_2 = c_2 r_2, \quad \chi = \frac{2\kappa}{\left|2 - c - \sqrt{c^2 - 4c}\right|} \tag{11.3}$$

The current location of particle i in dimension j at time t is represented as $x_{(i,j,t)}$, with the current velocity $v_{(i,j,t)}$. φ_1 and φ_2 act as random weightings for the

personal and neighbourhood bests, represented as $p_{(i,j,t)}$ and $p_{(g,j,t)}$ respectively. c_1 and c_2 are constants, usually set at 2.05, with $c = c_1 + c_2$. r_1 and r_2 are uniform random numbers in the range $[0, 1]$. Equation (11.3) calculates χ, a constant friction on the particles that prevents them from oscillating violently around an optimum. κ is usually set at 1.

11.2.3 Speciated Particle Swarms

Most PSO algorithms are limited in that they will only converge on a single solution, even when there are many global optima. Locating several solutions is beneficial in several ways. Firstly, it provides the user with a choice. While to the algorithm it may appear that two optima are of equal fitness, in reality it may be preferable to choose one over the other. In many environments there are factors that are too complex to incorporate into the fitness function. These factors are nevertheless present and may lead an expert user to choose one solution over another. Secondly, by simultaneously locating multiple solutions we reduce the risk of premature convergence, that is where the entire population becomes trapped in a local optimum.

Speciation, also known as niching, is one way to achieve this. The population is divided into species, which are groups of particles that are close to each other in the decision space. Communication between species is either severely limited or nonexistant, allowing them to each explore their local area without interference from particles on distant peaks.

We have used SPSO [21] as our base algorithm to test the regression method. To determine the effectiveness on dynamic environments we will be using Moving Peaks, a well-known dynamic test function generator (see section 11.4.2). SPSO is an ideal candidate for this function for two reasons. Not only does it perform well on dynamic multimodal functions [29], its performance is also strongly correlated with its local convergence speed [6]. This shows that by increasing the local convergence speed, we should see a marked improvement in overall performance.

In SPSO, each species is defined by a hyper-spherical area of radius r. This area is centred on the locally-fittest particle, called the species seed. Any particles within the species area are considered to belong to that species, although they are free to leave should they move away. The particles within each species are connected using the global neighbourhood topology [19]. There is no communication between particles of different species. To allocate the particles to species, they are first sorted from fittest to least-fit. The list is then iterated through; for each particle, if there is a species seed within r of its location it joins that species. Otherwise it becomes the seed particle of a new species. If a particle is within r of two or more species seeds it is allocated to the species of the fittest seed, as shown in Fig. 11.1.

In the original SPSO algorithm [21], particles were allocated to species based on their current location. To improve species stability we have used the personal best location and fitness, as was done in [5].

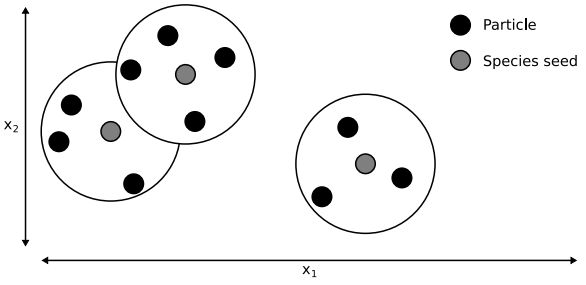


Fig. 11.1 The species seeds are the fittest particles in their area of the decision space. The middle seed is fitter than the one on the left, so its species area takes precedence where they overlap

11.2.4 Guaranteed Convergence PSO

The basic PSO model has an inefficiency, in that the velocity of the fittest particle will quickly drop to zero. This is caused by its personal and neighbourhood bests being on the same point. While this is not a big problem in the standard implementation, it can become noticeable when using speciated algorithms because the population of each species is often very low. Having an idle particle in a population of 20 is far less noticeable than in a population of 4. In order for the particle to start moving again, one of its neighbours must locate a better point. If there are only 2 or 3 particles this can take a long time, if it happens at all.

Guaranteed Convergence PSO [30] was developed to overcome this problem. Instead of travelling around like the rest of the population, the fittest particle randomly tries points within a distance d of its personal best location. GCPSO adaptively determines the value of d for each particle by tracking how many consecutive successful or unsuccessful tries there were. An attempt is considered successful if it improves on the personal best, otherwise it is a failure. If the number of consecutive successes exceeds a threshold, the algorithm searches more aggressively by doubling d. Likewise if the number of consecutive failures becomes too large d is halved so as to search in a smaller area. Combining SPSO with GCPSO means that the species seed will follow the GCPSO rules, ensuring that it never stops searching. The other particles follow the standard PSO implementation.

11.2.5 mQSO

To provide a benchmark to test SPSO's performance against, we have used one of the most effective PSO-based algorithms on Moving Peaks, mQSO [8]. mQSO modifies the standard PSO in several ways to improve performance on this function. These improvements are discussed below.

To track as many peaks as possible, mQSO divides the population into subswarms. These are equivalent to species in SPSO, except that particles are not free to join or leave a subswarm. If two subswarms become too close to each other, the weaker one will have its particles reinitialised. This prevents duplication and encourages the swarm to explore new areas. Stagnation is prevented by means of an anticonvergence measure. If all of the subswarms have converged to small areas, the weakest one is reinitialised in the same way as a duplicate subswarm. This prevents the system from wasting its resources on peaks of low fitness.

mQSO is used to increase the swarm's responsiveness to a peak movement. Rather than letting all of the particles tightly converge on the optimum, half of the particles are reserved as quantum particles. These particles do not follow the standard PSO movement equations; instead at each timestep they are placed randomly within a hypersphere of radius r_{cloud} using a uniform volume distribution. This technique is similar in some respects to GCPSO mentioned above.

11.3 Using Regression to Locate Optima

On a multimodal fitness landscape, around each peak there is a catchment area. Within this area, fitness generally improves as you get closer to the peak. If we can model the overall shape of the peak while ignoring the local features, we can calculate the highest point of that shape. Assuming that our model is reasonably accurate, the top of our shape should be close to the optimum. We can use regression to approximate the shape of the peak. Polynomial regression with least-squares approximation has been used to improve traditional optimisation methods [12, 35]. And more recently, regression was also employed to improve the performance of EAs [23, 32, 37].

In our proposed hybrid PSO using regression, we maintain a separate memory to the base algorithm, storing only the best locations and their fitnesses. If the base algorithm's memory was used, points would only remain known as long as there is an individual there. The regression needs to know the locations of the fittest points, regardless of the population's current state.

A minimum number of points is needed in order to calculate the regression – below this there will be more than one shape that fits the data. As the algorithm continues sampling the fitness landscape, the regression may keep some excess points to help reduce the effect of any local landscape features. The number of excess points e is a tunable, although robust, parameter.

By performing a linear least-squares regression on the known points and their fitnesses, we are able to estimate the peak's shape. From the regression we obtain a set of equations, one for each decision variable, that defines the shape that best fits our known points. Although more complex and flexible equations can be used if desired, for simplicity and efficiency we have used quadratic equations to represent the shape. This results in a set of simultaneous equations in the form of Equation (11.4) to be solved for $a_1, a_2, \ldots, a_{2n}$ and c, where n is the number of decision variables.

$$f(x_1,x_2,...,x_n) = a_1x_1^2 + a_2x_1 + a_3x_2^2 + a_4x_2 + \ldots + a_{(2n-1)}x_n^2 + a_{(2n)}x_n + c \tag{11.4}$$

In matrix form, the simultaneous equations look like:

$$\mathbf{A} = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_m \end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix} x_{1,1}^2 & x_{1,1} & x_{1,2}^2 & x_{1,2} & \ldots & x_{1,n}^2 & x_{1,n} & 1 \\ x_{2,1}^2 & x_{2,1} & x_{2,2}^2 & x_{2,2} & \ldots & x_{2,n}^2 & x_{2,n} & 1 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ x_{m,1}^2 & x_{m,1} & x_{m,2}^2 & x_{m,2} & \ldots & x_{m,n}^2 & x_{m,n} & 1 \end{bmatrix} \qquad \mathbf{C} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_{2n} \\ c \end{bmatrix}$$

where there is an equation for each of the m known points. To solve the simultaneous equations, we manipulate the matrices as in Equation (11.5):

$$\begin{aligned} \mathbf{A} &= \mathbf{BC} \\ \mathbf{B}^+\mathbf{A} &= \mathbf{B}^+\mathbf{BC} \\ \mathbf{B}^+\mathbf{A} &= \mathbf{C} \end{aligned} \tag{11.5}$$

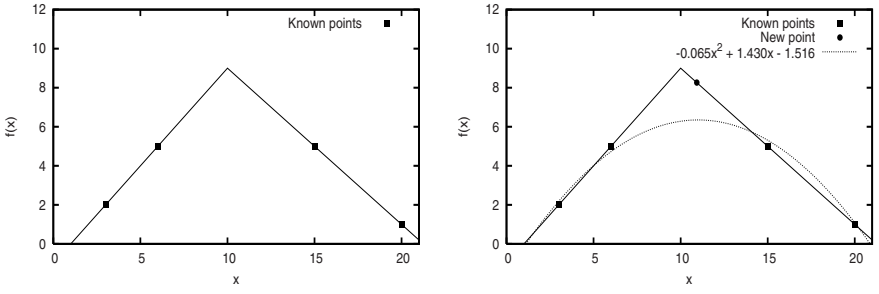
where $\mathbf{B}^+$ is the pseudoinverse of $\mathbf{B}$ [3]. If we only use the minimum number of points, $\mathbf{B}$ will be square and we can use the inverse $\mathbf{B}^{-1}$ instead. The minimum number of points required to perform an approximation is $2n+1$, according to Equation (11.4) (see also [23]). The coefficients that make our equation best match the known points are found by computing $\mathbf{C}$. We then find the turning point for the equation in each dimension $i = [1,n]$ by taking the partial derivative, as in Equation (11.6):

$$\frac{\partial}{\partial x_i} f(x_1,x_2,\ldots,x_n) = 2x_i a_{(2i-1)} + a_{(2i)} \tag{11.6}$$

The turning point in dimension i is where $\frac{\partial}{\partial x_i} f(x_1,x_2,\ldots,x_n) = 0$. To find out whether it is a maximum or minimum point, we take the second derivative. When using quadratic equations, we can simply look at the sign of $a_{(2i-1)}$; a negative number indicates that it is a maximum. If this is a maximisation problem and one of the equations has only a minimum turning point, we abort the regression and wait for better data. Similarly, if it is a minimisation problem we abort if any of the equations has no minimum turning point.

The global maximum point of the shape will be at the location of the turning point in each decision variable. Even though we were able to compute a maximum, we still need to check that it is valid. If the points do not give a good representation of the peak, for example they are all on one side, the regression will not be accurate. If the computed point is outside the expected area, or even the entire decision space, it is discarded. We will try the regression again when we have more data.

To test the calculated position, we replace the least-fit individual with a new individual at the shape's highest point. This avoids using an extra evaluation, and it is unlikely that the individual's next movement would have contributed much to the search. If the regression was successful, the new point will be used to further



(a) (b)

Fig. 11.2 a) Trying to find the highest point of the peak. We currently know the fitnesses of 4 points: 3, 6, 15 and 20. The right side of the peak is less steep than the left. b) The regression curve has a maximum at $x = 10.926$, considerably closer to the peak than any of the previously known points

refine the shape when it is next performed, hopefully improving the fitness still further. When using this technique with dynamic environments, we clear the memory whenever a peak movement is detected. This prevents the regression from being performed on stale data.

The main cost of this method is in performing the matrix inversion. Assuming the minimum number of points are used, this has a complexity of $\mathcal{O}(n^3)$. As n is dependent only on the number of decision variables and complexity of the equations used, the cost is usually quite low. The CPU cost can be further reduced by only performing the regression at certain intervals or only for the most promising peaks. In many environments, fitness evaluations are the most expensive aspect. The regression's minimal CPU overhead is usually far outweighed by the number of evaluations saved.

As an example we will try to solve a 1-dimensional triangular function, as shown in Fig. 11.2 a). Currently we know the fitnesses of 4 points:

$$\begin{aligned} f(3) &= 2 \\ f(6) &= 5 \\ f(15) &= 5 \\ f(20) &= 1 \end{aligned}$$

We place these values into $\mathbf{B}$ and $\mathbf{C}$:

$$\begin{bmatrix} 2 \\ 5 \\ 5 \\ 1 \end{bmatrix} = \begin{bmatrix} 3^2 & 3 & 1 \\ 6^2 & 6 & 1 \\ 15^2 & 15 & 1 \\ 20^2 & 20 & 1 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ c \end{bmatrix}$$

Multiplying both sides by $\mathbf{B}^+$ gives:

$$\begin{bmatrix} 0.01 & -0.01 & -0.01 & 0.01 \\ -0.24 & 0.12 & 0.30 & -0.17 \\ 1.46 & 0.02 & -1.01 & 0.54 \end{bmatrix} \begin{bmatrix} 2 \\ 5 \\ 5 \\ 1 \end{bmatrix} = \begin{bmatrix} a_1 \\ a_2 \\ c \end{bmatrix}$$

$$\begin{bmatrix} -0.065 \\ 1.430 \\ -1.516 \end{bmatrix} = \begin{bmatrix} a_1 \\ a_2 \\ c \end{bmatrix}$$

This gives us the best-fitting quadratic curve, Equation (11.7).

$$f(x) = -0.065x^2 + 1.430x - 1.516 \tag{11.7}$$

To find the turning point, we differentiate it:

$$\frac{df(x)}{dx} = (-0.065)2x + 1.430 \tag{11.8}$$

Solving Equation (11.8) gives a turning point of $x = 10.926$. We know this point is the maximum because the x^2 coefficient is negative. The fitness at $x = 10.926$ is 8.2592. As can be seen from Fig. 11.2 b), this is not the location of the actual peak, however it is considerably closer than any of the points known so far.

There are similarities between EANA [23] and the technique presented here. Both methods use a polynomial regression to estimate the location of a peak. However there are also important differences, including the focus of the algorithms. EANA is designed to estimate the area of the global optimum, whereupon local search is used to refine its guess. As with all local search algorithms, premature convergence can be a problem. Our technique takes advantage of the fact that modern EAs are already very effective at locating the area of a peak. Instead we use the regression as a heuristic, guiding the base algorithm towards its goal as it explores the surrounding areas. This gives us the best of both worlds – substantially improving performance while only minimally increasing the risk of premature convergence.

Another important difference is the way the two techniques obtain an accurate model of the landscape. EANA assumes that the height of the local optima is correlated with their distance from the global optimum, and so spends evaluations searching for the local peaks. Our technique makes no such assumption, instead using more than the required number of points to compute its model. This allows it to better reflect the general trends of the landscape and discourages overfitting, without needing additional evaluations.

11.4 Experimental Setup

To determine whether performing the regression is effective, we compared the performance of SPSO and GCPSO with and without the regression. For the rest of the paper, we will use SPSO to mean SPSO + GCPSO, and rSPSO to mean SPSO +

GCPSO + regression. The regression has been implemented so as to discard any calculated solutions that are outside the species boundary, as described in Section 11.3.

For the purposes of the regression, we consider each species to be an individual subpopulation with its own memory. This means that for every timestep, there is a regression run for each species.

The regression will be tested on both static and dynamic multimodal test functions; we will describe our procedure below. For all of the tests, each species is limited to $P_{max} = 6$ particles; any excess particles are reinitialised elsewhere in the decision space. This is the same method as was used in [29] to avoid having too many individuals crowd an optimum. The success and failure thresholds for GCPSO have been set to the values recommended in [4], that is $s_c = 15$ and $f_c = 5$. Unless otherwise stated, the regression stores a maximum of $e = 10$ excess points. Each experiment was performed 50 times and the results have been averaged.

11.4.1 Static Functions

To test general performance in a static multimodal environment, we chose functions that represents several different landscape features. These functions are described below; the mathematical definitions are shown in Table 11.1.

- Inverted Branin RCOS (F1) and Himmelblau (F4) both have peaks with large catchment areas.
- Six-Hump Camel Back (F2) has two global optima with relatively large catchment areas, however there are also 4 local optima for the particles to become trapped in.
- Deb's 1st Function (F3) has 5 narrow narrow peaks; even though it is only 1 dimensional, it can be difficult to locate all of the optima.

Table 11.1 Static multimodal test functions

Function	r	**Comments**
Inverted Branin RCOS [9]: $F1(x,y) = -[(y - \frac{5x^2}{4\pi^2} + \frac{5x}{\pi} - 6)^2 + 10(1 - \frac{1}{8\pi})cos(x) + 10]$, where $-5 \le x \le 10$; $0 \le y \le 15$	4	3 global optima
Six-Hump Camel Back [25]: $F2(x,y) = -4[(4 - 2.1x^2 + \frac{x^4}{3})x^2 + xy + (-4 + 4y^2)y^2]$, where $-1.9 \le x \le 1.9$; $-1.1 \le y \le 1.1$	1	2 global optima and 4 local optima
Deb's 1st Function [13]: $F3(x) = sin^6(5\pi x)$, where $0 \le x \le 1$	0.15	5 equally spaced global optima
Himmelblau [2]: $F4(x,y) = 200 - (x^2 + y - 11)^2 - (x + y^2 - 7)^2$, where $-6 \le x,y \le 6$	3	4 global optima
Inverted Shubert 2D [20]: $F5(x,y) = -\sum_{i=1}^{5} i\cos[(i+1)x + i] \sum_{i=1}^{5} i\cos[(i+1)y + i]$, where $-10 \le x,y \le 10$	0.75	18 global optima in 9 clusters, many local optima

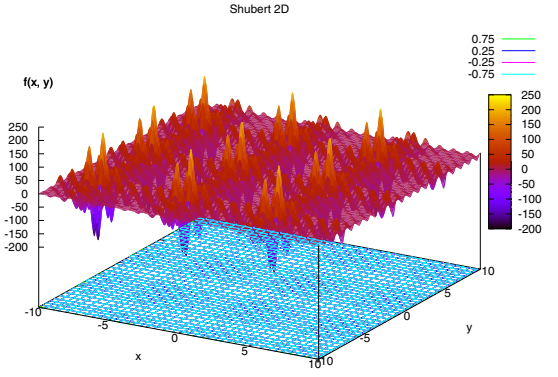


Fig. 11.3 The inverted Shubert 2D test function has 18 global optima located in 9 pairs, as well as many local optima

- Inverted Shubert 2D (F5) is the most difficult as it is highly multimodal. There are 18 global optima, all of which have extremely small catchment areas. In addition, there are numerous local optima as shown in Fig. 11.3.

A run is only considered successful if the algorithm locates all of the optima to within a fitness of $\varepsilon = 0.00001$ within 2000 timesteps.

For F1 through F4, a population size of 50 has been used. To reliably solve F5, SPSO requires at least 500 particles. With the exception of Deb's First function, all of these are two dimensional functions.

In addition to the test functions in Table 11.1, we also used a modified version of the Generic Hump Function proposed by Singh and Deb [33], to test the regression's performance in environments where there are a larger number of decision variables. The Hump function allows us to independently control the number of dimensions, the number of humps and the size and shape of those humps, making it ideal for our testing.

The Humps function consists of K humps rising from a flat surface. The height of the hump k at a given point X is shown in Equation (11.9):

$$f(x,k) = \begin{cases} h_k \left[1 - \left(\frac{d(X,k)^{\alpha_k}}{r_k}\right)\right], & \text{if } d(X,k) < r_k \\ 0, & \text{otherwise} \end{cases} \tag{11.9}$$

The distance from the centre of the hump k to X is denoted by $d(X,k)$. In Singh's testing, all of the humps have the same height, radius and shape; $h_k = 1, \alpha_k = 1$ for all k. The radius r_k was varied with the number of dimensions. The humps are placed so that they do not intersect, meaning that the distance between any two humps j and k is at least $r_j + r_k$. For our testing, we have removed the function's flat base so as to allow SPSO to more easily find the peaks, as shown in Equation (11.10):

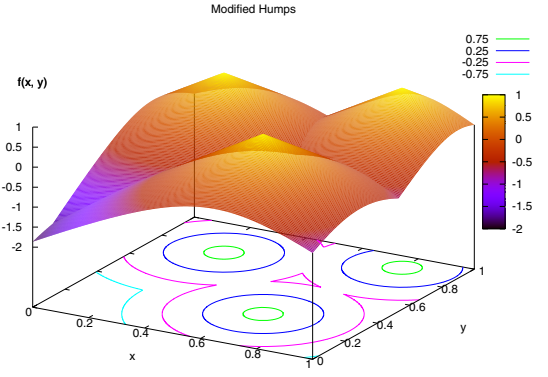


Fig. 11.4 An example of the Modified Humps landscape using 3 humps in two dimensions

$$f(x,k) = h_k \left[1 - \left(\frac{d(X,k)^{\alpha_k}}{r_k} \right) \right] \tag{11.10}$$

The height of any given point in the decision space is the height of the tallest hump at that point, as in Equation (11.11):

$$f(x) = \mathrm{Max}_{k=1}^{K} f(x,k) \tag{11.11}$$

An example of a landscape with 3 humps in two dimensions is shown in Fig. 11.4. We have tested this function with 20 peaks and 5, 10, 15 and 20 decision variables, with r_k set to 0.27, 0.37, 0.41 and 0.43 respectively. All of the tests used 300 particles and SPSO's r parameter was set to $2r_k$, that is twice the hump radius.

11.4.2 Moving Peaks

Moving Peaks is a highly configurable dynamic test function suite, and is a common benchmark for dynamic optimisation algorithms [10]. Moving Peaks is very similar to the Humps function described above, except that the peaks periodically move and change size. The peaks are usually conic, however in this paper we will be testing with other shapes as well.

The algorithm must track the peaks as they move around the fitness landscape. As the peaks move they also change height; this means that the globally optimal peak changes over time, as in Fig. 11.5. For this reason, algorithms that only track one or a few peaks tend to perform poorly; the results of a particular run depend more on the peak's average height than the algorithm's performance [6]. To achieve good performance it is critical that an algorithm tracks as many peaks as possible. It is too expensive to locate the new optimum from scratch after every landscape change. Following this rationale, it is expected that the good performance of an optimiser for

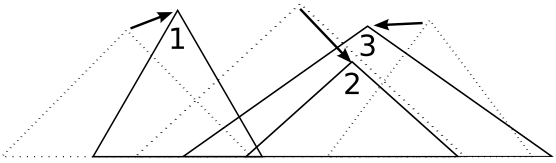


Fig. 11.5 The difficulty of tracking peaks: Peak 2 used to be the global optimum however it is now covered by Peak 3. At some point in the future Peak 2 may re-emerge and become the highest again; algorithms with insufficient population diversity are unlikely to rediscover this peak

a static multimodal environment should be transferable to the dynamic multimodal environment.

Unless otherwise stated, all Moving Peaks parameters are set as specified by scenario 2; please refer to Table 11.2 for details[1].

The most widely-used performance metric on Moving Peaks function is offline error [10]. Offline error is the difference in fitness between the best-known point and the global optimum, averaged over the entire run. All of our results on Moving Peaks will be presented in terms of offline error after 500,000 evaluations.

Both the PSO and regression algorithms need to be informed when the landscape changes. To detect these changes, at the end of each timestep we check the fitness of the top 5 species seeds. If any of the fitnesses differ from the recorded value, the particles' personal best memories and the memory used for the regression are both cleared.

To determine the robustness of the regression under different circumstances, we tested:

- The number of peaks between 1 and 200
- The severity of each peak movement, between 0 and 6
- The number of decision variables between 5 and 10

11.4.2.1 Creating Varying Peak Shapes

Extending our preliminary work in [7], we will fully analyse the regression's performance, including on more complex peak shapes, as defined in this section. We wanted to see whether the regression was still effective with other peak shapes. The standard conic shape used by Moving Peaks is produced by Equation (11.12):

$$f(x) = h - w\sqrt{\sum_{d=1}^{D}(x_d - p_d)^2} \tag{11.12}$$

[1] See also: http://www.aifb.uni-karlsruhe.de/∼jbr/MovPeaks/

Table 11.2 Moving Peaks Scenario 2 parameters

Parameter	Setting
Random seed	1
Dimensions	5
Peaks	10
Minimum peak height	30
Maximum peak height	70
Standard peak height	50
Minimum peak width	1
Maximum peak width	12
Standard peak width	0
Coordinate range	[0, 100]
Peak movement severity	1
Peak height severity	7
Peak width severity	1
Basis function	None
Movement correlation λ	0
Peak movement interval	5000 evaluations
Peak shape	Conic
Change stepsize	Constant

where w and h specify the width and height of the peak respectively, x_d is the location of the point in dimension d and p_d is the tip of the peak in dimension d. We have extended this equation in several ways:

- The relationship between height and distance from the peak's centre in dimension d is now controlled by α_d. $\alpha_d = 1$ will produce the conic shape used by the standard Moving Peaks scenarios. Using $\alpha_d > 1$ will produce a mound shape, with steepness increasing as the α_d gets bigger. Setting $\alpha_d \in (0,1)$ produces a spike shape, as depicted in Fig. 11.6.
- To create asymmetric peaks, for each decision variable we divide the peak into 2 halves, left and right. The value of α_d used for the left and right halves are denoted α_{d0} and α_{d1} respectively. Fig. 11.7 shows an asymmetric peak.
- Local optima have been added by superimposing a cosine wave over the peak. The amplitude and frequency of the wave are specified by β_d and γ_d respectively. All γ values used in this paper are in radians. Fig. 11.8 shows a cosine wave superimposed on Fig. 11.7. By adjusting β_d and γ_d we can specify the number and severity of local peaks in variable d.

The new peak function is defined in Equation (11.13). The standard conic peak shape can be achieved by setting $\alpha_d = 1, \beta_d = 0, \gamma_d = 0$ for all d.

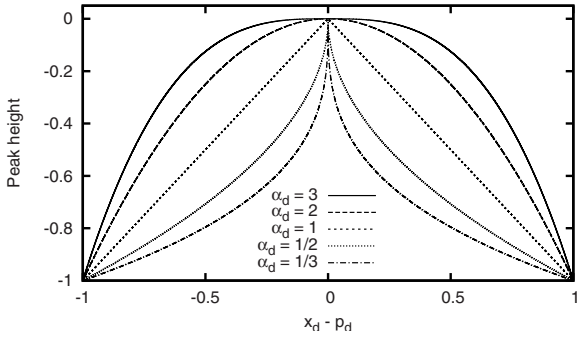


Fig. 11.6 Five peaks have been overlaid, showing how α_d determines the peak's shape

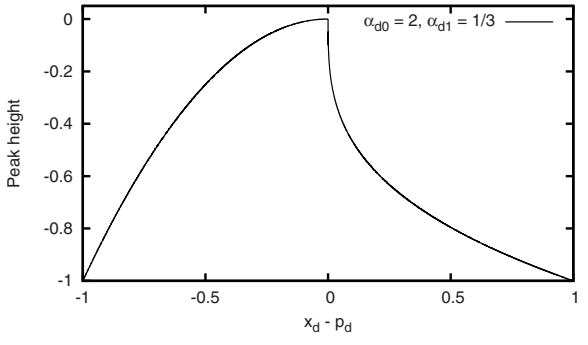


Fig. 11.7 An example of an asymmetric peak. The values of α_d are 2 and $\frac{1}{3}$ for the left and right halves respectively

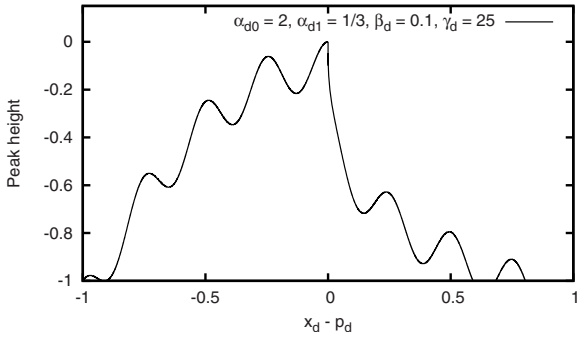


Fig. 11.8 Superimposing the cosine wave over the peak in Fig. 11.7

$$f(x) = h - w\sqrt{\sum_{d=1}^{D}\left(|x_d - p_d|^{\alpha_d} + u_d\right)^2} \tag{11.13}$$

where

$$\alpha_d = \begin{cases} \alpha_{d0}, \text{ if } x_d < p_d \\ \alpha_{d1}, \text{ otherwise} \end{cases}, u_d = \beta_d\left(cos\left[\gamma_d\left(x_d - p_d\right)\right] - 1\right), \alpha_{d0}, \alpha_{d1} \in (0,\infty), \beta_d, \gamma_d \in [0,\infty)$$

To determine the regression's performance on varying peak types, the following tests were carried out:

- Symmetric peaks where all α_{d0} and α_{d1} values are equal. We tested with the following setting: $\alpha_d \in \left\{\frac{1}{3}, \frac{1}{2}, 1, 2, 3\right\}, \beta = \gamma = 0$.
- Asymmetric peaks where all α_d values are randomly generated within the range $[\frac{1}{3}, 3]$, and $\beta = \gamma = 0$.
- Conic peaks with a superimposed wave: $\alpha_d = 1, Max_\beta \in [2, 10], Max_\gamma \in [20, 100]$.
- Asymmetric peaks with a superimposed wave: $\alpha_{d0}, \alpha_{d1} \in [\frac{1}{3}, 3]$, $Max_\beta \in [2, 10]$, $Max_\gamma \in [20, 100]$.

Max_β indicates that each peak's values for β_d are randomly chosen in the range $[0, Max_\beta]$. Max_γ indicates the same thing for γ.

We also tested the regression's sensitivity to e, both on the standard scenario 2 problem and with the modified peak function, using $\alpha_d \in [\frac{1}{3}, 3], Max_\beta = 10, Max_\gamma = 100$. The latter is designed to show whether using a larger e value improves performance on functions with many local optima.

Finally we compared our results to mQSO, one of the best performing algorithms on Moving Peaks scenario 2. We have compared our results to the $10\,(5 + 5^q)$ configuration that Blackwell showed to be optimal on this problem. All experiments were run with 100 particles and with SPSO's r parameter set to 30. These were the settings used in [22].

11.5 Results

This section is divided into two main parts. We will first look at the regression's performance on static multimodal functions. Secondly we will analyse its behaviour on the Moving Peaks test suite.

11.5.1 Static Functions

Even with static environments, it is still very important to reduce the number of evaluations. Each evaluation costs CPU time, often well in excess of the time used by the optimiser itself. In this part we will report on the regression's performance on static problems, both in low dimensional and high dimensional environments. The

Table 11.3 Regression performance on low dimensional static functions. Mean, standard error and improvement over SPSO are shown

Function	rSPSO	SPSO	Improvement
F1	6254.54 (±298.59)	9552.86 (±216.98)	35%
F2	1489.63 (±53.90)	8934.58 (±304.97)	83%
F3	5306.60 (±225.23)	7613.16 (±271.40)	30%
F4	4963.74 (±277.75)	11069.68 (±299.91)	55%
F5	50511.46 (±794.94)	164360.00 (±2912.89)	69%

Table 11.4 Regression performance on the high dimensional Humps function. d is the number of decision variables

d	rSPSO	SPSO	Improvement
5	191902.50 (±5353.82)	292472.57 (±12638.88)	34%
10	256795.42 (±2506.12)	356665.10 (±5146.87)	28%
15	277200.14 (±2038.94)	404640.00 (±1161.45)	31%
20	297978.00 (±1427.11)	462036.00 (±1216.41)	36%

third part of this section will investigate how the number of excess points affects performance.

11.5.1.1 Low Dimensional Landscapes

Table 11.3 shows that using the regression dramatically increased performance on all of the low dimensional functions we tested. The largest improvement was on Branin RCOS, where the number of evaluations required was reduced by more than 80%. Even on Deb's First Function (F3), the regression still reduced the number of evaluations by nearly a third. This is a significant improvement.

Inverted Shubert 2D (depicted in Fig. 11.3) was by far the hardest 2 dimensional function we tested, normally requiring 160,000 evaluations for SPSO to locate all of the optima. Adding the regression reduced this to just 50,000. We suspect that one of the reasons the regression performed so well is that the peak tips resemble a parabola, allowing it to be accurately modelled.

11.5.1.2 High Dimensional Landscapes

The number of dimensions does not appear to affect the regression's effectiveness. Table 11.4 shows that using the regression reduced the number of evaluations by about 30%. This becomes especially signficant as the number of dimensions

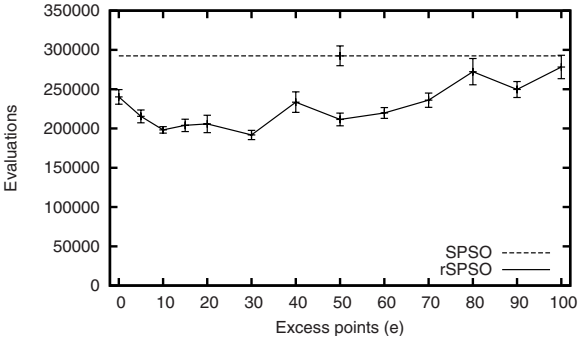


Fig. 11.9 Number of evaluations needed for Humps 5D with different numbers of excess points. The vertical bars show one standard error

increases; for the 20 dimensional function using the regression saved over 160,000 evaluations.

11.5.1.3 Sensitivity to *e*

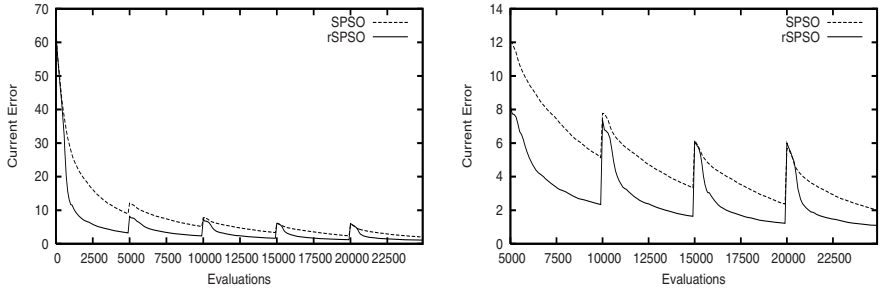
As Fig. 11.9 shows, the best performance was obtained by setting *e* to between 10 and 30. Although it still beat SPSO, the regression performed relatively poorly when excess points were not kept. The Humps function's peaks are conic, making them difficult to model when only using the minimum number of points. The excess points help to define the surface better, improving the regression's guess.

Using too many points also decreased performance. The regression does not perform any weighting – it tries to match all of the points regardless of their fitness. By storing too much data, we allow the memory to become polluted with distant and poor quality points. Instead of just modelling the tip, the peak's overall shape is matched. We are then unable to accurately determine the tip's location as our model is not specific to that area.

We recommend that *e* be set to 10 for all problems. This value represents a good tradeoff; it provides excellent performance without creating too much CPU or memory overhead. We have also tested this on the Moving Peaks problem and found the same ideal value. This is discussed in the next section.

11.5.2 Moving Peaks

Reducing the number of evaluations is critical when working with dynamic environments. If the environment is changing every 2 minutes, an algorithm that takes 3 minutes to find an adequate solution is useless. By adding the regression we are able to significantly reduce the number of evaluations needed; this section details the performance on Moving Peaks under different situations.



(a) (b)

Fig. 11.10 Current error over time on Moving Peaks scenario 2; a) showing the effect of adding the regression; b) the same as a) but without the first 5000 evaluations

11.5.2.1 Increasing Convergence Speed

Fig. 11.10 a) compares the current error over time for SPSO and rSPSO. Current error is the difference in fitness between the best known point and the global optimum. Offline error is calculated by averaging the current error over an entire run.

In scenario 2, the peaks move every 5000 evaluations, as can be seen by the upwards jumps in the graphs. The regression is inactive for at least the first 200 evaluations after a peak movement. As the population size is 100, this represents only two timesteps; improvements here are mostly due to the fortunate placement and existing momentum of the particles. The regression cannot be used yet because there are insufficient points for it to be computed. For a 5-dimensional function, 11 points are needed. Since each species is limited to 6 particles, at least two timesteps needed to collect the required data.

Fig. 11.10 b) is the same as a), but without the first 5000 evaluations. This gives a better indication of the regression's performance helping to track the peaks. After a peak movement the PSO requires 100 evaluations to re-evaluate the particles, one for each individual. Thus the indicated error immediately after a change is not representative of the individuals' true fitnesses.

At 300 evaluations after a peak movement the regression's effect becomes obvious. The curve for rSPSO drops quickly as the algorithm hones in on the optimum. After a landscape change, a normal PSO must wait for the particles to accelerate towards the new peak, then wait for them to converge again once the peak has been located. The combination of GCPSO and the regression reduces the time spent doing this: the regression moves the worst particle close to the peak while GCPSO's rules set the velocity to 0 and force it to explore the local area.

At 1200 evaluations after a peak movement rSPSO has already achieved the same error as SPSO does after 4900 evaluations. Even at this point, the curve has a fairly steep gradient; substantial improvements are being made each timestep. The graph plateaus around 3000 evaluations; the error at this point is very small. The curve does not converge to 0 because SPSO is usually unable to maintain species on all of

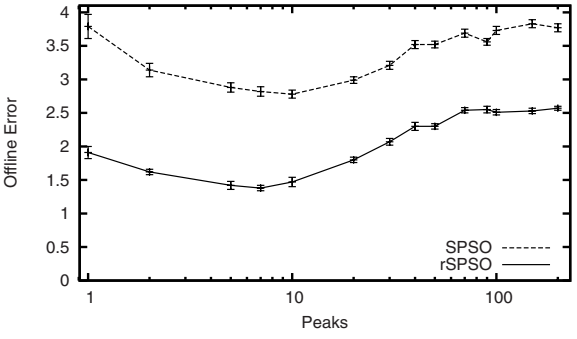


Fig. 11.11 Adding the regression substantially reduced offline error

the optima [6]; the residual error is from the times that the algorithm is not tracking the highest peak.

Please note that to ensure a smooth curve for Fig. 11.10, we have performed 1000 runs for each of the algorithms. All other results presented in this paper are based on experiments of 50 runs.

11.5.2.2 Number of Peaks

Adding the regression reduced offline error by between 1 and 1.5, as shown in Fig. 11.11. The settings of SPSO ($r = 30, P_{max} = 6$) are optimised for 10 peaks, which explains the sweet spot at that point. Below this, there are too many particles for the number of peaks. Since only 6 particles are allowed on each peak, when there are too few peaks most of the particles are continually reinitialised. This wastes evaluations, resulting in a larger offline error.

At the other end of the graph it becomes impossible for the algorithm to track all of the peaks. Having neither enough particles nor a small enough r value, the algorithm must rely on the particles to jump between peaks, hopefully to the globally optimal one. The increased error is caused by the times the algorithm is unable to discover the best peak.

11.5.2.3 Peak Movement Severity

The peak movement severity controls how far the peak moves each time. Larger severities increase the time needed to re-find the peak. The further the peaks move, the faster the particles will be travelling when they reach it. In a standard PSO they will then take longer to slow down and reconverge. Using the regression in combination with GCPSO helps this process; whenever the regression's guess is successful, the worst particle will become the species seed as the regression's guess was better than any of the existing solutions. Since the species seed follows the GCPSO movement rules, it immediately loses the momentum it previously had.

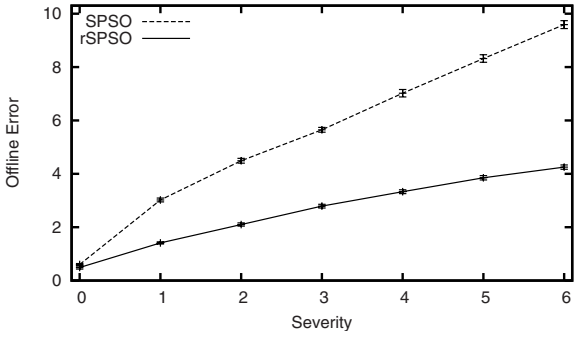


Fig. 11.12 The benefit of the regression increases with severity

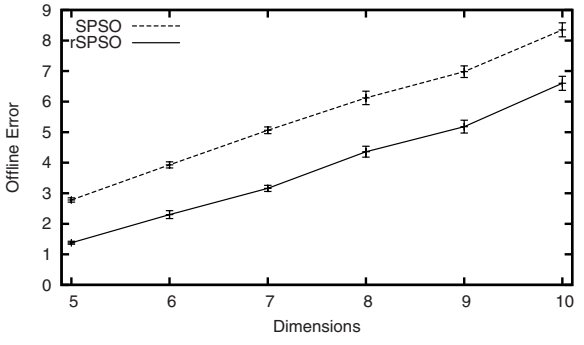


Fig. 11.13 The regression reduced the offline error by around 1.5 for all of the dimension values tested

As this happens to successive particles, the average velocity is quickly reduced, allowing the particles to reconverge.

Fig. 11.12 shows how SPSO's error increases linearly with severity. rSPSO's error also increases, but at a much lower rate. At a severity of 0 the peaks do not move at all, they only change height. In this situation the error achieved depends almost entirely on being able to track all of the peaks. As the peaks are not moving, once the optima have been initially located the regression is not needed to track them, thus the performances of SPSO and rSPSO are very similar.

11.5.2.4 Dimensionality

As with the results shown in Table 11.4 for the Hump function, performance improvement provided by the regression on the Moving Peaks is fairly constant. The error increases linearly as the number of dimensions increases, however the difference between rSPSO and SPSO remains about 1.5, as shown in Fig. 11.13.

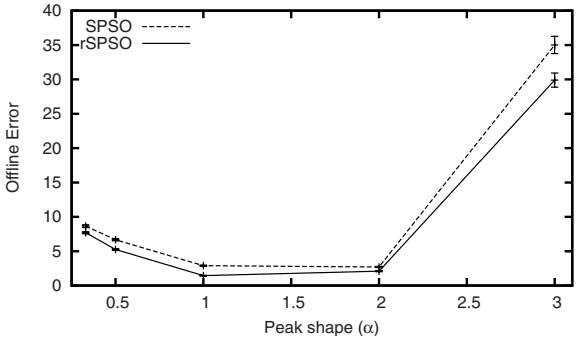


Fig. 11.14 Offline error for different peak shapes

11.5.2.5 Peak Shape

For this experiment, all of the α_d values have been set identically, making the shapes tested symmetrical. A one dimensional peak for each value of α_d is depicted in Fig. 11.6.

The most obvious feature of Fig. 11.14 is that outside the range $\alpha_d \in [1,2]$ there is a large increase in offline error.

For $\alpha_d > 2$, the areas away from the optima are exceptionally steep. Performance depends almost entirely on how quickly the algorithm can locate the general area of the tip, rather than its exact location. Guiding the PSO's search, the regression is able to reduce the offline error from 35 to 30.

When $\alpha_d < 1$, the penalty for being far away from the peak's tip is relatively small, however to achieve a small error it is extremely important to precisely locate the optimum. The difference between a point 0.1 units away from the peak and 0.2 units away can be substantial. Again, using the regression reduced the offline error by about 1. This is quite impressive as the actual peak shape is the opposite of the regression's model – the fitness landscape does not fit a parabola at all. This result suggests the parabolic model works well even on difficult peak shapes, and that more elaborate models are generally unnecessary. Further testing would be required to confirm this though.

11.5.2.6 Asymmetric Peaks

In the real world, many fitness landscapes have asymmetrical peaks. The peak may be very steep on one or more sides, or be at the edge of the feasible region. For these experiments, we have used random peak shapes. The α_d values for each side of every peak in each dimension are chosen randomly within a specified range. For example, in a run with only two peaks, the following values may be chosen:

$$\alpha_{00} = 0.40, \alpha_{01} = 0.94, \alpha_{10} = 2.64, \alpha_{11} = 0.58$$

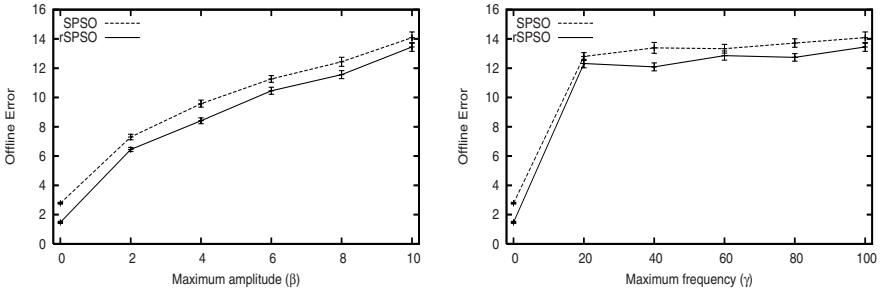
Table 11.5 Regression performance on asymmetric peaks

α_d	rSPSO	SPSO	Improvement
$[0.33, 3]$	1.82 ($\pm$0.07)	2.59 ($\pm$0.09)	30%
$[0.4, 2.5]$	1.70 ($\pm$0.07)	2.46 ($\pm$0.07)	31%
$[0.5, 2]$	1.66 ($\pm$0.06)	2.67 ($\pm$0.07)	38%
$[0.67, 1.5]$	1.58 ($\pm$0.05)	2.66 ($\pm$0.09)	41%
$[1, 1]$	1.45 ($\pm$0.05)	2.90 ($\pm$0.08)	50%

As Table 11.5 shows, the regression is most effective when the range of α_d is small. The regression works better on peaks that are roughly symmetrical as they more closely match the parabolic shape used. However even when highly asymmetrical peaks are created, the regression still achieved a 30% improvement over SPSO. This shows again that the regression's guesses are still accurate enough to aid the optimisation, even when it cannot closely model the peak.

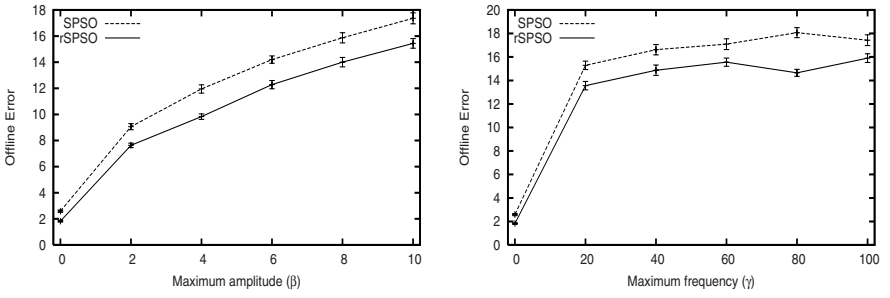
11.5.2.7 Adding Local Optima

By comparing Fig. 11.15 a) and b) we can see that the wave's amplitude had a much greater effect on performance than its frequency. The flatness of Fig. 11.15 b) suggests that the swarm is able to jump from peak to peak with relative ease. The amplitude's effect is far greater because each new candidate solution is just as likely to be at the bottom of the wave as the top. On average each new point will be halfway down a wave, increasing the overall error incurred. The local optima decrease the accuracy of the regression's model, reducing its effectiveness. Even so, it still managed to reduce the offline error by around 1 in all of the runs.



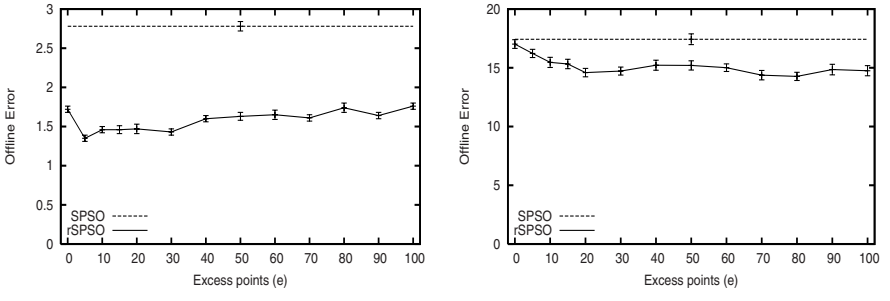
(a) (b)

Fig. 11.15 Performance on symmetric peaks for different values of: a) Max_β ($\alpha_d = 1, Max_\gamma = 100$); b) different values of Max_γ ($\alpha_d = 1, Max_\beta = 10$)



(a) (b)

Fig. 11.16 Performance on asymmetric peaks for different values of: a) Max_β ($\alpha_d \in [0.33, 3]$, $Max_\gamma = 100$); b) Max_γ ($\alpha_d \in [0.33, 3]$, $Max_\beta = 10$)



(a) (b)

Fig. 11.17 a) Offline error for Moving Peaks Scenario 2 for different values of e. b) Offline error for asymmetric peaks with waves for different values of e ($\alpha_d \in [0.33, 3], Max_\beta = 10, Max_\gamma = 100$)

11.5.2.8 Asymmetric Peaks with Local Optima

This is the most challenging landscape for the PSO; we are creating peaks that look similar to the one shown in Fig. 11.8. In all of the experiments the regression reduced the offline error by between 1.5 and 2. As can be seen by comparing Fig. 11.16 a) and b) with the results for the asymmetric peaks and local optima individually, the local optima are the primary cause of the large offline error - the asymmetric peaks are not a significant component. As would be expected, the results here are very similar to the results for the local optima tests.

11.5.2.9 Sensitivity to e

On scenario 2 the value of e does not greatly affect the regression's performance. Low, nonzero values provided slightly better results for the same reason as before,

Table 11.6 Comparing against mQSO: Severity

s	mQSO	rSPSO	SPSO
0	1.18 (±0.07)	0.49 (±0.06)	0.60 (±0.04)
1	1.75 (±0.06)	1.41 (±0.04)	3.02 (±0.07)
2	2.40 (±0.06)	2.10 (±0.06)	4.49 (±0.09)
3	3.00 (±0.06)	2.79 (±0.07)	5.65 (±0.09)
4	3.59 (±0.10)	3.33 (±0.07)	7.02 (±0.14)
5	4.24 (±0.10)	3.85 (±0.08)	8.32 (±0.14)
6	4.79 (±0.10)	4.25 (±0.08)	9.59 (±0.15)

Table 11.7 Comparing against mQSO: Number of peaks

Peaks	mQSO	rSPSO	SPSO
1	5.07 (±0.17)	1.91 (±0.09)	3.79 (±0.18)
2	3.47 (±0.23)	1.62 (±0.04)	3.14 (±0.10)
5	1.81 (±0.07)	1.42 (±0.06)	2.88 (±0.07)
7	1.77 (±0.07)	1.38 (±0.04)	2.82 (±0.07)
10	1.80 (±0.06)	1.47 (±0.07)	2.78 (±0.06)
20	2.42 (±0.07)	1.80 (±0.04)	2.99 (±0.05)
30	2.48 (±0.07)	2.07 (±0.05)	3.21 (±0.06)
40	2.55 (±0.07)	2.30 (±0.06)	3.52 (±0.06)
50	2.50 (±0.06)	2.30 (±0.04)	3.52 (±0.05)
100	2.36 (±0.04)	2.51 (±0.04)	3.73 (±0.06)
200	2.26 (±0.03)	2.57 (±0.03)	3.77 (±0.06)

by allowing the regression to concentrate on the best information. In all cases the regression outperformed SPSO by a significant margin, as shown in Fig. 11.17 a).

When optimising the most complex function, Moving Peaks with $\alpha_d \in [0.33, 3]$, $\beta = 10, \gamma = 100$, the value of e played a larger role in performance (Fig. 11.17 b)). As would be expected, increasing e slightly helped the regression ignore the local optima. Values larger than 20 gave no extra advantage however, showing that even for difficult problems only a few excess points are needed.

11.5.2.10 Comparing to mQSO

In Table 11.6 we compare the performance against mQSO (with the anticonvergence measure) for differing movement severities. As can be seen, rSPSO exceeds mQSO's performance for all of the severities tested. This is even more impressive considering that for most of the experiments SPSO had far worse performance

than mQSO. It should also be noted that both mQSO and SPSO have been tuned for this benchmark - the parameters chosen by Blackwell were optimised for each severity setting. SPSO's r has been set to the standard value for the Moving Peaks benchmark. The only parameter specifically related to the regression, e, requires very little tuning. The value of 10 was shown to be either optimal or near-optimal for all of the tests we conducted.

Table 11.7 compares mQSO's performance against both SPSO and rSPSO for differing numbers of peaks. It can be seen that rSPSO is highly competitive with mQSO; offering better performance for all but the 100 and 200 peak runs. It should be remembered that the regression can be added to most numerical optimisation algorithms; it is highly likely that it could be used to improve mQSO's performance even further.

11.6 Conclusion

In this chapter we have presented a technique to incorporate regression into a PSO algorithm, in order to improve local convergence. We have provided experimental studies and analysis of results using several multimodal test functions with varying difficulty. We have also extended the Moving Peaks test suite with more complex peak shapes, and carried out experimental studies on these newly defined test functions. Our results show that the performance of the regression-based SPSO (rSPSO) compares favorably against two existing multimodal PSOs (SPSO and mQSO). In particular, adding the regression significantly improved the performance of an existing multimodal PSO algorithm (SPSO) on a range of fitness landscapes in both static and dynamic environments. By using the existing population members, the regression technique does not require any additional evaluations, but only a modest amount of memory and CPU time depending on the number of decision variables and excess points.

As a future research direction it may be worthwhile exploring other deterministic techniques that could be combined with a PSO. Currently much of the available information is thrown away as the population moves each generation. By retaining and analysing this data, it is likely that further improvements in performance can be found.

References

1. Barthelemy, J.F.: Approximation concepts for optimimum structural design - a review. Structural Optimization 5, 129–144 (1993)
2. Beasley, D., Bull, D., Martin, R.: A sequential niche technique for multimodal function optimization. Evolutionary Computation 1(2), 101–125 (1993)
3. Ben-Israel, A., Greville, T.: Generalized Inverses: Theory and Applications. Wiley-Interscience, New York (1974)
4. van den Bergh, F., Engelbrecht, A.: A new locally convergent particle swarm optimiser. In: Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, vol. 3, pp. 96–101. IEEE Press, Hammamet (2002)

5. Bird, S., Li, X.: Enhancing the robustness of a speciation-based PSO. In: Proceedings of the Congress on Evolutionary Computation, pp. 843–850. IEEE Press, Vancouver (2006)
6. Bird, S., Li, X.: Informative performance metrics for dynamic optimisation problems. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 18–25. ACM Press, London (2007)
7. Bird, S., Li, X.: Using regression to improve local convergence. In: Proceedings of the Congress on Evolutionary Computation, pp. 1555–1562. IEEE Press, Singapore (2007)
8. Blackwell, T., Branke, J.: Multi-swarm optimization in dynamic environments. Applications of Evolutionary Computing, 489–500 (2004)
9. Branin, F.: Widely convergent method for finding multiple solutions of simultaneous nonlinear equations. IBM Journal of Research and Development 16(5), 504–522 (1972)
10. Branke, J.: Memory enhanced evolutionary algorithms for changing optimization problems. In: Proceedings of the Congress on Evolutionary Computation, vol. 3, pp. 1875–1882. IEEE Press, Washington (1999)
11. Clerc, M., Kennedy, J.: The particle swarm - explosion, stability, and convergence in a multidimensional complex space. IEEE Transactions on Evolutionary Computation 6, 58–73 (2002)
12. Conn, A., Toint, P.: An algorithm using quadratic interpolation for unconstrained derivative free optimization. Nonlinear Optimization and Applications, 27–47 (1996)
13. Deb, K., Goldberg, D.: An investigation of niche and species formation in genetic function optimization. In: Proceedings of the International Conference on Genetic Algorithms, pp. 42–50. Morgan Kaufmann, San Francisco (1989)
14. Gottfried, B.: Introduction to Optimization Theory. Prentice-Hall, Englewood Cliffs (1973)
15. Jin, Y.: A comprehensive survey of fitness approximation in evolutionary computation. Soft Computing-A Fusion of Foundations, Methodologies and Applications 9(1), 3–12 (2005)
16. Jin, Y., Olhofer, M., Sendhoff, B.: Managing approximate models in evolutionary aerodynamic design optimization. In: Proceedings of IEEE Congress on Evolutionary Computation, vol. 1, pp. 592–599 (2001)
17. Jin, Y., Olhofer, M., Sendhoff, B.: A framework for evolutionary optimization with approximate fitness functions. IEEE Transactions on Evolutionary Computation 6(5), 481–494 (2002)
18. Kennedy, J., Eberhart, R.: Swarm intelligence. Morgan Kaufmann, San Francisco (2001)
19. Kennedy, J., Mendes, R.: Population structure and particle swarm performance. In: Proceedings of the Congress on Evolutionary Computation, pp. 1671–1676. IEEE Press, Honolulu (2002)
20. Li, J., Balazs, M., Parks, G., Clarkson, P.: A species conserving genetic algorithm for multimodal function optimization. Evolutionary Computation 10(3), 207–234 (2002)
21. Li, X.: Adaptively choosing neighbourhood bests using species in a particle swarm optimizer for multimodal function optimization. In: Deb, K., et al. (eds.) GECCO 2004. LNCS, vol. 3102, pp. 105–116. Springer, Heidelberg (2004)
22. Li, X., Branke, J., Blackwell, T.: Particle swarm with speciation and adaptation in a dynamic environment. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 51–58. ACM Press, Seattle (2006)
23. Liang, K.H., Yao, X., Newton, C.: Evolutionary search of approximated n-dimensional landscapes. International Journal of Knowledge-Based Intelligent Engineering Systems 4(3), 172–183 (2000)

24. Mahinthakumar, G., Sayeed, M.: Hybrid genetic algorithm - local search methods for solving groundwater source identification inverse problems. Journal of Water Resources Planning and Management 131(1), 45–57 (2005)
25. Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs. Springer, New York (1996)
26. Michalewicz, Z., Fogel, D.: How to Solve It: Modern Heuristics. Springer, Berlin (2000)
27. Neumaier, A.: Molecular modeling of protein and mathematical prediction of protein structures. SIAM Review 39(3), 407–460 (2002)
28. Ong, Y., Nair, P., Keane, A.: Evolutionary optimization of computationally expensive problems via surrogate modeling. AIAA Journal 41(4), 687–696 (2003)
29. Parrott, D., Li, X.: Locating and tracking multiple dynamic optima by a particle swarm model using speciation. IEEE Transactions on Evolutionary Computation 10(4), 440–458 (2006)
30. Peer, E., van den Bergh, F., Engelbrecht, A.: Using neighbourhoods with the guaranteed convergence PSO. In: Proceedings of the IEEE Swarm Intelligence Symposium, pp. 235–242. IEEE Press, Indianapolis (2003)
31. Ratle, A.: Accelerating the convergence of evolutionary algorithms by fitness landscape approximation. In: Eiben, A.E., Bäck, T., Schoenauer, M., Schwefel, H.-P. (eds.) PPSN 1998. LNCS, vol. 1498, pp. 87–96. Springer, Heidelberg (1998)
32. Runarsson, T.: Ordinal Regression in Evolutionary Computation. In: Runarsson, T.P., Beyer, H.-G., Burke, E.K., Merelo-Guervós, J.J., Whitley, L.D., Yao, X. (eds.) PPSN 2006. LNCS, vol. 4193, pp. 1048–1057. Springer, Heidelberg (2006)
33. Singh, G., Deb, K.: Comparison of multi-modal optimization algorithms based on evolutionary algorithms. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 1305–1312. ACM Press, Seattle (2006)
34. Vesterstrom, J., Riget, J., Krink, T.: Division of labor in particle swarm optimisation. In: Proceedings of the Congress on Evolutionary Computation, vol. 2, pp. 1570–1575. IEEE Press, Honolulu (2002)
35. Winfield, D.: Function minimization by interpolation in a data table. Journal of the Institute of Mathematics and its Applications 12, 339–347 (1973)
36. Yin, P., Yu, S., Wang, P., Wang, Y.: A hybrid particle swarm optimization algorithm for optimal task assignment in distributed systems. Computer Standards & Interfaces 28(4), 441–450 (2006)
37. Zhou, Z., Ong, Y., Nguyen, M., Lim, D.: A study on polynomial regression and Gaussian process global surrogate model in hierarchical surrogate-assisted evolutionary algorithm. In: Proceedings of the 2005 Congress on Evolutionary Computation, vol. 3, pp. 2832–2839. IEEE Press, Los Alamitos (2005)