Using Regression to Improve Local Convergence

Stefan Bird, *Member, IEEE*, and Xiaodong Li, *Member, IEEE*

Abstract— Traditionally Evolutionary Algorithms (EAs) choose candidate solutions based on their individual fitnesses, usually without directly looking for patterns in the fitness landscape discovered. These patterns often contain useful information that could be used to guide the EA to the optimum. While an EA is able to quickly locate the general area of a peak, it can take a considerable amount of time to refine the solution to accurately reflect its true location.

We present a new technique that can be used with most EAs. A surface is fitted to the previously-found points using a least squares regression. By calculating the highest point of this surface we can guide the EA to the likely location of the optimum, vastly improving the convergence speed. This technique is tested on Moving Peaks, a commonly used dynamic test function generator. It was able to significantly outperform the current state of the art algorithm.

I. Introduction

Gradient descent methods are known for their extremely quick convergence, however they are also highly susceptible to becoming trapped in the first local optimum they find. In addition, they are only usable when the slope of each point in the fitness landscape can be calculated. For many real world problems this is impossible as the model may not follow an obvious mathematical formula.

Evolutionary Algorithms (EAs) have also proved to be an effective search strategy. They are able to converge on an optimum even when the catchment area occupies only a small portion of the search space. An EA, especially when combined with a diversification measure, is far less likely to become trapped in a local optimum than gradient descent.

In this paper we present a new method that combines the advantages of both approaches, without requiring the gradient information to be known. Rather than using gradient of the function, we compute an n-dimensional surface that best fits the known points. We then attempt to locate the highest point of this surface. Provided that the local features of the fitness landscape roughly match our surface, the local 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.

To determine whether our method is effective or not, we have benchmarked it using the Moving Peaks [1] test suite. This function consists of a number of peaks that wander randomly around the decision space, the goal being to track the highest peak as closely as possible.

Stefan Bird is with the School of Computer Science and Information Technology, RMIT University, Melbourne, Victoria, Australia (email: stbird@seatiger.org).

Xiaodong Li is with the School of Computer Science and Information Technology, RMIT University, Melbourne, Victoria, Australia (email: xiaodong@cs.rmit.edu.au).

We have organised the paper as follows: Section II will introduce the existing methods that we will be using, as well as the current state of the art. Section III will describe our convergence enhancement, followed by Sections IV and V which detail our testing methodology and results respectively. Finally, we will conclude the paper in Section VI, giving some possible directions for future research.

II. Prior Work

This work adds a technique similar to the gradient descent method to existing EAs. For our testing, we have used a speciated PSO with a local convergence enhancer. These will be described in the following section.

A. Gradient Descent

Gradient descent methods work by using the derivative of the fitness function to guide the search direction [2]. The next point chosen to search is one that is close to the last point evaluated but in the direction of the steepest descent. For functions where we are attempting to maximise the fitness instead of minimise we simply reverse this - ie we find the direction of the slope up instead of down.

B. Particle Swarms

Particle Swarm Optimisation (PSO) models a flock of birds searching for a resource. Each bird is watching for changes in its neighbours' behaviour. Since in general an individual's behaviour does not change arbitrarily, it is likely that any new behaviour is due to previously unknown information. By copying the behaviour of its neighbours, an individual is able to benefit from the new information even before it has gained the information itself.

PSO operates on a similar principle. Each particle maintains its current location and velocity, and the location of the fittest point so far, known as its *personal best*. It also has a set of neighbours with whom it can communicate its personal best. To decide where to move at each timestep, the particle randomly chooses a point somewhere near its own personal best and the fittest personal best of any of its neighbours. It then adjusts its velocity towards this point while retaining some momentum from its previous velocity. This causes the particle to circle around the region, closely exploring the most promising areas.

To guarantee convergence, we have used Clerc's constriction coefficient PSO [3], [4]. This is described by Equations (1) and (2), which are run for every timestep t.

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

$$x_{(i,j,t+1)} = x_{(i,j,t)} + v_{(i,j,t+1)} \qquad (2)$$

where:

$$\varphi_1 = c_1 r_1, \quad \varphi_2 = c_2 r_2,$$

$$\chi = \frac{2\kappa}{\left|2 - c - \sqrt{c^2 - 4c}\right|} \qquad (3)$$

$x_{(i,j,t)}$ is the current location of particle i in dimension j, with the current velocity symbolised as $v_{(i,j,t)}$. φ_1 and φ_2 act as random weightings for the personal and neighbour 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 (3) calculates χ, a constant friction on the particles that prevents them from oscillating violently around an optimum. κ is traditionally set at 1.

C. Speciated Particle Swarms

Most PSOs in their basic form are limited in that they will converge on a single solution. There is no guarantee that the found solution is the global best; it may have prematurely converged on a local optimum, a situation from which it can rarely escape.

Speciation, also called niching, allows the swarm to return multiple potential solutions where they exist. As individuals in different areas of the decision space are prevented from interacting, they allow each species to locate its optimum without interference. By simultaneously converging on multiple solutions, the population is less likely to become trapped in a local optimum – even if one species becomes trapped, others are still able to search their area of the decision space.

For our experiments, we have used Li's SPSO algorithm [5] which provides good performance on the Moving Peaks test function [6]. This algorithm is ideal because of the high correlation between its local convergence speed and overall performance, showing that if we are able to improve the local convergence speed, we should see a significant performance improvement [7].

SPSO works by grouping the particles into species according to their location within the decision space. Particles that are the fittest in their region are considered species *seeds*. The rest of particles are then allocated to the species of the fittest seed particle in their local area. A particle's local area is described by a hypersphere centred on the particle's current location with a radius of r, a user-set parameter. This is shown in Figure 1.

Although the original SPSO presented used the current location and fitness of each particle to define species, to increase species stability we have used the personal best location and fitness as was done in [8].

D. Guaranteed Convergence PSO

One drawback of the basic PSO model is that the velocity of the fittest particle tends to zero very quickly. When the personal best and local best points are the same the particle will stop moving, regardless of whether it is on an actual
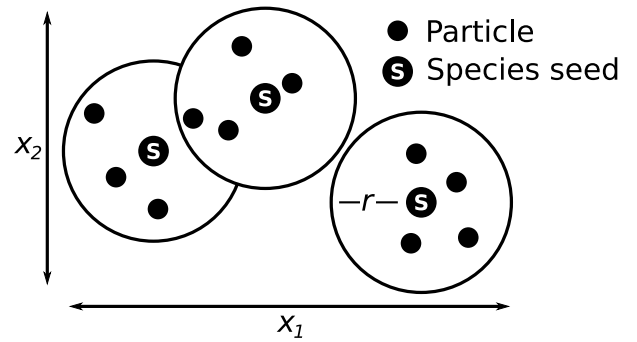

Fig. 1. The species seeds represent the fittest particles in their areas of the decision space. The middle seed is fitter than the one on the left, so its species area takes precedence where they overlap.

optimum or not. Another particle in its neighbourhood must find a better location before the particle will start moving again.

This problem is especially pronounced in speciated algorithms - since the population of each species is vastly smaller than the overall population, the portion of the search power each particle contributes is much greater. Having an inactive individual hurts a swarm of 5 particles far more than a swarm of 50 particles.

The Guaranteed Convergence PSO (GCPSO) [9] overcomes this problem by applying different movement rules to the fittest particle within each neighbourhood. Instead of moving around normally as the remaining particles do, the fittest particle randomly tries points within a certain distance d of its personal best location. The value of d is determined adaptively for each particle. As the particle tries its surrounding points, it keeps track of the number of times in a row it was successful or not in finding a fitter location. If the number of successes in a row exceeds a threshold, the particle becomes more aggressive by doubling d. Likewise if the number of failures reaches a threshold it halves d, thereby searching in a smaller area.

By implementing SPSO in combination with GCPSO, we can ensure that the species seed - by definition the particle in the best position to locate the optimum - continues searching effectively.

E. Moving Peaks

The Moving Peaks test suite [1] is a highly configurable fitness evaluator, and is a commonly used benchmark for dynamic optimisation algorithms. It consists of a number of peaks which move around the decision space, changing height and width as they do so. For simplicity, a conic peak shape is generally used, although this is by no means a requirement.

The most commonly used performance metric for this suite is *offline error* [1], calculated by the accumulated error between the fitness of the best known point and the actual global optimum, divided by the number of evaluations. This problem is challenging because of the nature of the peak movement – since the peaks are changing in both height and
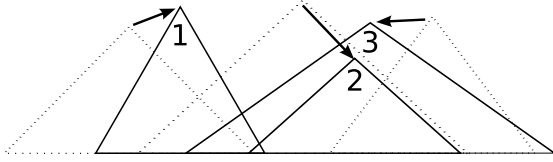
Fig. 2. The difficulty of tracking peaks: Peak 2 used to be the global optimum however it is now covered by Peak 3. At some point in the future Peak 2 may re-emerge and become the highest again; algorithms with insufficient population diversity are unlikely to rediscover this peak.

location, a currently sub-optimal peak may become the global best at some point in the future, as in Figure 2. To achieve good performance it is necessary to both track as many peaks as possible and reconverge as quickly as possible after a peak movement.

Recently we proposed a new metric, BKPE [7], that exclusively measures the exploitation aspect of an algorithm. This metric reports how quickly an algorithm converges on the best peak it knows about, without penalising it for choosing a non-global optimum. It is calculated in a similar way to offline error, except that we keep track of the error for all peaks, rather than just the global one. Immediately before a landscape change, we choose the peak with the fittest individual that was known about since the last landscape change. We then add this peak's error to our total error for the run.

F. Multiswarms with QSO

The current state of the art in performance on Moving Peaks is mQSO – Blackwell and Branke's combination of multiswarms with QSO [10]. To achieve this, a standard PSO was modified in several ways to optimise its performance on this benchmark.

Firstly they divided the population into several subswarms to allow the PSO to converge on multiple optima simultaneously. This has an effect similar to using SPSO, however the subswarms (species in SPSO) are defined beforehand rather than being based on the population. Should two subswarms move too close, the particles in the less fit swarm will be reinitialised – it will be forced to find a new peak.

The subswarms are subject to an anticonvergence measure: if all of the subswarms have converged below a given radius, the least fit one is killed and its particles reinitialised. This prevents the system from wasting resources on an underperforming peak and becomes more and more critical as the number of peaks increases.

Finally, to allow the PSO to react more quickly to peak movements, a quantum swarm is used. The population is divided into quantum and normal particles. At each iteration, the quantum particles are moved randomly within a hypersphere of radius r_{cloud} according to a uniform volume distribution. The hypersphere is centred on the fittest known point. The rest of the particles follow the normal particle movement equations (1) and (2).

III. USING REGRESSION TO LOCATE OPTIMA

On a multimodal landscape, each peak has its own catchment area. Allowing for local noise and local optima, points within the catchment area will generally get fitter as we move towards the peak. We can take advantage of this property by assuming the peak has a certain shape and calculating where the top of that shape should be.

The concept of local convergence should be emphasised here. For this enhancement to be effective, the points used must all belong to the same peak. If points from neighbouring peaks are used the algorithm will not provide any benefit, and may even degrade performance slightly. In general, an EA will naturally converge onto a single peak, so with enough time this becomes a non-issue.

We maintain a list of the fittest known points, which is separate to the current population - that is a location's fitness is remembered even if there is no longer an individual actively representing it. To calculate the shape of the local peak we perform a linear least squares regression on the fittest known points. We represent the shape as a set of equations (one for each decision variable) that satisfy as closely as possible the fitnesses of our known points. For simplicity we have used quadratic equations, although more complex and flexible equations can be used if desired. This results in a set of equations in the form of Equation (4) to be solved simultaneously for $a_1, a_2, ..., a_{2n}$ and c, where n is the number of decision variables.

$$f(x_1, x_2, ..., x_n) = a_1 x_1^2 + a_2 x_1 + a_3 x_2^2 + a_4 x_2 + \ldots$$
$$+ a_{(2n-1)} x_n^2 + a_{(2n)} x_n + c \quad (4)$$

In matrix form, the simultaneous equations look like:

$$\mathbf{A} = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_m \end{bmatrix}$$

$$\mathbf{B} = \begin{bmatrix} x_{1,1}^2 & x_{1,1} & \cdots & x_{1,n}^2 & x_{1,n} & 1 \\ x_{2,1}^2 & x_{2,1} & \cdots & x_{2,n}^2 & x_{2,n} & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ x_{m,1}^2 & x_{m,1} & \cdots & x_{m,n}^2 & x_{m,n} & 1 \end{bmatrix}$$

$$\mathbf{C} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_m \\ c \end{bmatrix}$$

Where there is an equation for each of the m known points.

$$\mathbf{A} = \mathbf{BC}$$
$$\mathbf{B^+A} = \mathbf{B^+BC}$$
$$\mathbf{B^+A} = \mathbf{C} \quad (5)$$

$\mathbf{B}^+$ is the pseudoinverse of $\mathbf{B}$. If we only use the minimum number of points, $\mathbf{B}$ will be square and we can use the inverse $\mathbf{B}^{-1}$ instead. By computing $\mathbf{C}$ we determine the coefficients that make our equation best match the known points. We then find the turning point for the equation in each dimension $i = [1, n]$ by taking the partial derivative, as in Equation (6):

$$\frac{\partial}{\partial x_i} f(x_1, x_2, \ldots, x_n) = 2x_i a_{(2i-1)} + a_{(2i)} \qquad (6)$$

The turning point in dimension i is where $\frac{\partial}{\partial x_i} f(x_1, x_2, \ldots, x_n) = 0$. To find whether it is a maximum or minimum point we take the second derivative, although in the case of a quadratic equation we can simply look at the sign of $a_{(2i-1)}$ – if it is negative the point is a maximum. If the turning point in any of the decision variables is a minimum, we abort attempting to calculate the peak location and wait for better data before trying again.

The global maximum point of the shape will be at the location of the turning point in each decision variable. A sanity check is required here – if the points are ill-spaced around the optimum the regression may be inaccurate. If the calculated maximum is outside the decision space it is discarded; we simply try again next generation where hopefully we will have better data. As we will be combining this method with SPSO, in our experiments we discard any calculated maximum that is further than r from the species seed, ensuring that any newly discovered point will still belong to the same species.

Once the maximum has been calculated, the current least fit individual representing the peak is moved to this point. If the calculated location becomes one of the fittest points, it will be used to compute the regression next generation, hopefully improving the fitness even more.

When a peak movement is detected, the known points for each peak are cleared. This prevents the system from performing the regression using stale information.

As an example we will try to solve a 1-dimensional triangular function, as shown in Figure 3. Currently we know the fitnesses of 4 points:

$$f(3) = 2$$
$$f(6) = 5$$
$$f(15) = 5$$
$$f(20) = 1$$

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

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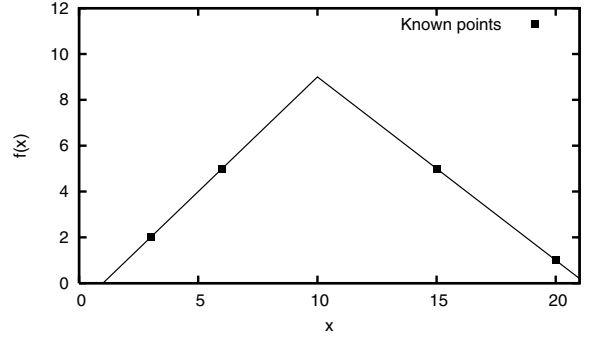

Fig. 3. Trying to find the highest point of the peak. We currently know the fitnesses of 4 points - 3, 6, 15 and 20. The right side of the peak is less steep than the left.
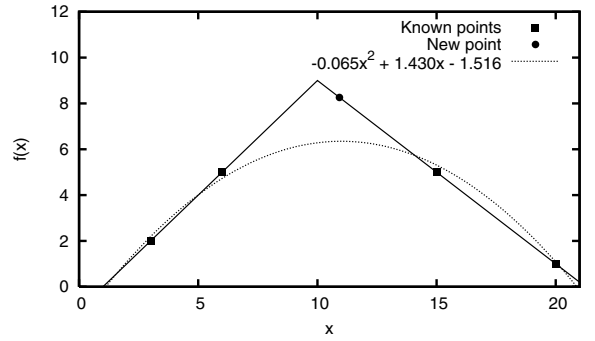

Fig. 4. The regression curve has a maximum at $x = 10.926$, considerably closer to the peak than any of the previously known points.

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 (7).

$$f(x) = -0.065x^2 + 1.430x - 1.516 \qquad (7)$$

To find the turning point, we differentiate it:

$$\frac{df(x)}{dx} = (-0.065)2x + 1.430 \qquad (8)$$

Solving Equation (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 Figure 4 this is not the location of the actual peak, however it is considerably closer than any of the points known so far.

The computational cost of this method primarily comes from the matrix inversion, which has a complexity of $O(n^3)$, assuming we use the minimum number of points. Thus the

cost is dependent only on the number of decision variables, which is usually quite low. By performing the regression only on certain timesteps or for certain species we can easily reduce the computational cost to meet a time budget. In many environments performing a fitness evaluation is the most expensive aspect, meaning that the modest CPU cost of this method is considerably outweighed by the number of evaluations saved.

IV. EXPERIMENTAL SETUP

To determine whether performing the regression is effective, we compared the performance of SPSO + GCPSO with and without the regression heuristic. For simplicity and to better show the effect of the regression, we did not use the anticonvergence measure discussed in [11]. As the technique we are presenting here is essentially SPSO with a regression heuristic, we will name it rSPSO. The setup is the same as was used in [11]: the total population is 100 particles with a species radius $r = 30$. Each species was limited to $P_{max} = 10$ particles with any excess being reinitialised with a random location and velocity. This technique was shown to improve performance by Parrott and Li in [6]. For GCPSO, we set the success and failure thresholds to the values recommended in [12], that is $s_c = 15$ and $f_c = 5$ respectively.

The regression does introduce one new parameter, that is how many excess points e to use. Using more points can potentially increase the accuracy of the regression, however it comes at the cost of slightly higher CPU and memory consumption. We have tested using only the minimum number of points required to perform the regression (in this case $2n+1$ where n is the number of decision variables), keeping every point with a known fitness, and keeping only a set number of excess points.

For the Moving Peaks setup, scenario 2 was used so as to allow direct comparison with other papers. For each experiment we performed 50 runs of 500000 evaluations. Unless otherwise stated, we used 10 peaks in a 5 dimensional decision space. The peak severity was set at 1 and the peaks were moved every 5000 evaluations. To detect peaks movements, at the end of each generation we checked the fitnesses of the top 5 species seeds. If any of the fitnesses differ from the recorded value, the personal best memory of each particle is reset to the current location. In addition the points used for the regression are cleared.

To determine the effect of the various Moving Peaks parameters, we have tested:

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

V. RESULTS

In this section we will be discussing the algorithm's sensitivity to the various parameters. Firstly we will discuss the effect of the maximum excess parameter e, the only parameter that this method adds. Subsequently we will
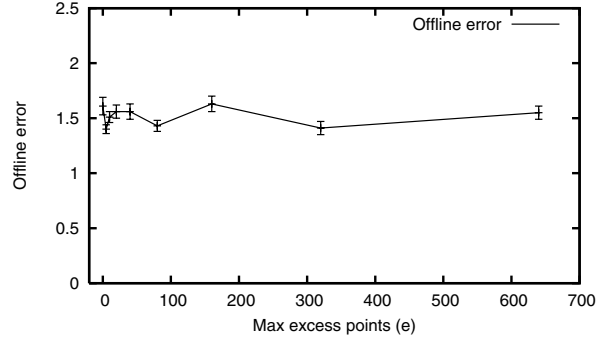
Fig. 5. The effect of e. In our tests, increasing the number of points in the regression did not improve performance.

analyse the performance as we change various aspects of the Moving Peaks benchmark, changing the difficulty in various ways. Finally we will compare our results with mQSO, using the $10(5+5^q)$ configuration that Blackwell et. al. showed to be optimal on this problem.

A. Sensitivity to e

In our tests, we saw no significant effect caused by the value of e. As can be seen in Figure 5, while there is some variation in the offline error achieved, there was no overall trend. Since each species is limited to 10 particles and the peaks are moved every $\frac{5000 \text{ evaluations}}{100 \text{ particles}} = 50$ iterations, the maximum number of points any species could accumulate before the next peak movement is 500, thus anything above this represents infinity.

It is likely that the conic form of the peaks is too regular to benefit much from having more regression points. We expect that functions with many local fluctuations will benefit more from having excess points, although we have not as yet explored this hypothesis. It is possible that using more flexible equations for the regression would yield better results than simply adding more points, although we risk overfitting the surface by doing this.

B. The effect of dimensionality

As shown in Figure 6, the number of dimensions does not greatly affect the relative benefit of the regression. Figure 7 shows that the improvement in BKPE was mirrored by a similar improvement in offline error, between approximately 1 and 1.5 in all of the tests. Similar mirroring between offline error and BKPE was seen in all of our experiments; for brevity in our subsequent tests we will only report the offline error. It can also be seen from this test that while using the regression does improve performance, it is still heavily reliant on the underlying algorithm's reaction to fitness landscape changes.

As the peaks are conic, the regression can only determine the optimum once it knows a point on both sides in every dimension. If, in any of the dimensions, all of the points are on the same side there is no way to compute the regression; to do so would require fitting a quadratic curve to points on a straight line, a mathematical impossibility. The regression
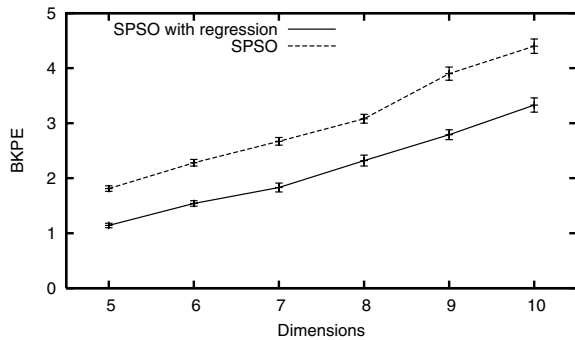
Fig. 6. While the number of dimensions had an approximately linear effect on the BKPE, the difference between SPSO with and without the regression was largely constant.

Fig. 7. The reduction in BKPE for rSPSO is mirrored by the offline error. Improving SPSO's convergence speed resulted in significantly better overall performance.
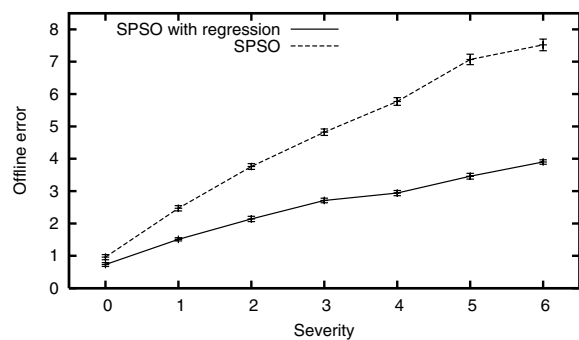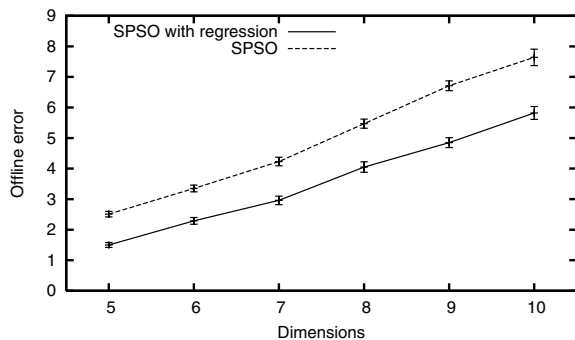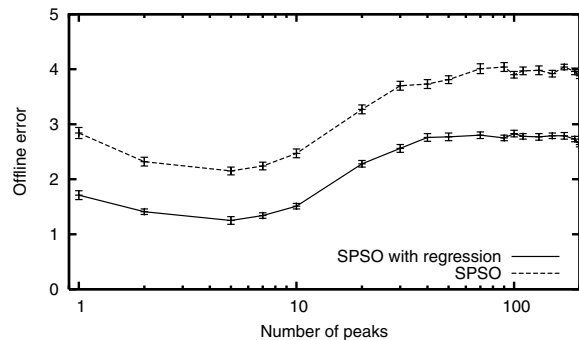

Fig. 8. The performance gain from using the regression increases as the peak movements become more severe. This data is also presented in Table I.

Fig. 9. The sweet spot occurs when the population size is roughly equal to P_{max} times the number of peaks. This data is also presented in Table II.

must wait until the particles have deconverged enough to locate the other side of the peak; this takes longer as the number of dimensions increases.

C. Peak movement severity

For a normal PSO there are two distinct stages after a peak movement. First, the swarm's particles gradually increase speed as they deconverge, caused by the discovery of better and better points as they move up the slope to the new peak – the attractor points, and thus force, will always in approximately the same direction from the particle. The second stage is to reconverge once the new location has been found, losing the speed they had previously gained.

Figure 8 shows that using the regression is most effective when the severity is larger. The further the new optimum is from the previous one, the faster the particles will be moving when they reach it, and the longer it will take to slow down again. When using the regression the new peak can be accurately located without waiting for the particles to reduce their velocity – as soon as a point on each side of the peak is known we can estimate the surface. Each successful estimation provides an additional point which helps refine future regressions. If the estimated peak happens to become the fittest-known point, the least fit particle will become the species seed. This causes it to follow the GCPSO rules, immediately losing whatever velocity it previously had. As

this happens to successive particles the overall velocity is rapidly reduced, lessening the time needed to reconverge.

D. Changing the number of peaks

As with the changing number of dimensions, the effect of adding the regression was a reasonably constant offline error reduction of between 1 and 1.5 (Figure 9). With a population size of 100 SPSO can only adequately represent 30 to 50 peaks – beyond this most peaks would have only 1 particle on them, giving very poor convergence. While the offline error increased up until about 70 peaks, after this it plateaued because the peaks were becoming close enough that the particles can easily move from one to the other.

As we have 100 particles and a maximum of 10 per species, it is impossible for the system to have all of the particles near an optimum when there are less than 10 peaks. As multiple species approach the same peak, the number of particles in the species will exceed the limit, causing the worst particles to be reinitialised. These particles will then restart their climb on the nearest peak, only to be reinitialised again once they get too close to an existing species. Reinitialising excess particles does benefit population diversity when there are undiscovered or under-represented peaks, however in this case there are no unknown peaks left to "discover". Throwing particles around costs evaluations while doing nothing to aid diversity or local convergence, thus we see poor offline-error performance in

TABLE I

COMPARISON OF RSPSO AND MQSO WITH CHANGING SEVERITY.
FIGURES SHOW AVERAGE AND STANDARD ERROR OF OFFLINE ERROR
PERFORMANCE.

s	mQSO (AC)	rSPSO	SPSO
0	1.18 ±0.07	0.74 ±0.08	0.95 ±0.08
1	1.75 ±0.06	1.50 ±0.08	2.51 ±0.09
2	2.40 ±0.06	1.87 ±0.05	3.78 ±0.09
3	3.00 ±0.06	2.40 ±0.08	4.96 ±0.12
4	3.59 ±0.10	2.90 ±0.08	5.56 ±0.13
5	4.24 ±0.10	3.25 ±0.09	6.76 ±0.15
6	4.79 ±0.10	3.86 ±0.11	7.68 ±0.16

TABLE II

COMPARISON OF RSPSO AND MQSO WITH DIFFERENT NUMBERS OF
PEAKS. FIGURES SHOW AVERAGE AND STANDARD ERROR OF OFFLINE
ERROR PERFORMANCE.

Peaks	mQSO (AC)	mQSO (no AC)	rSPSO	SPSO
1	5.07 ±0.17	5.07 ±0.17	1.42 ±0.06	2.64 ±0.10
2	3.47 ±0.23	3.47 ±0.23	1.10 ±0.03	2.31 ±0.11
5	1.81 ±0.07	1.81 ±0.07	1.04 ±0.03	2.15 ±0.07
7	1.77 ±0.07	1.77 ±0.07	1.21 ±0.05	1.98 ±0.04
10	1.80 ±0.06	1.75 ±0.06	1.50 ±0.08	2.51 ±0.09
20	2.42 ±0.07	2.74 ±0.07	2.20 ±0.07	3.21 ±0.07
30	2.48 ±0.07	3.27 ±0.11	2.62 ±0.07	3.64 ±0.07
40	2.55 ±0.07	3.60 ±0.08	2.76 ±0.08	3.85 ±0.08
50	2.50 ±0.06	3.65 ±0.11	2.72 ±0.08	3.86 ±0.08
100	2.36 ±0.04	3.93 ±0.08	2.93 ±0.06	4.01 ±0.07
200	2.26 ±0.03	3.86 ±0.07	2.79 ±0.05	3.82 ±0.05

this situation. As the number of peaks decreases the over-abundance increases, raising offline error.

E. Comparing rSPSO with mQSO

Table I shows the comparison between rSPSO and mQSO for different peak severities. As can be seen, rSPSO was able to clearly outperform mQSO on all of the severities tested. To our knowledge, mQSO was the best performing algorithm for Moving Peaks scenario 2, showing that rSPSO is a significant advance in the state of the art.

It should also be noted that mQSO has been tuned for this benchmark - the parameters chosen by Blackwell et. al. are optimised for each severity setting, and for Moving Peaks in general. SPSO has also been tuned to a limited degree – r has been set to 30, the standard value when optimising this benchmark. The only parameter specifically related to the regression, e, has not undergone tuning – it was set at 10 for every run. Section V-A shows it to have had little to no effect on performance for this function.

When testing with different numbers of peaks, Blackwell et. al. also reported the effect of the anticonvergence measure, showing the significant improvement in performance. Table II compares SPSO and rSPSO to mQSO, with and without the anticonvergence (AC) measure. When there are fewer than 20 peaks, rSPSO outperforms mQSO as indicated above and in Table I. Above this rSPSO slightly trails mQSO, however it still performs significantly better than mQSO without the anticonvergence.

The table also shows that mQSO scales better to large numbers of peaks than SPSO. The difference between mQSO with and without the anticonvergence measure is quite large, suggesting that SPSO may be being penalised by a relative inability to "jump ship" from a low peak to a higher one nearby. Further research would be required to determine whether this is actually the case or not.

Overall it can be seen that rSPSO is able to outperform mQSO in most cases. This is especially impressive given that the only performance enhancement the two algorithms share is non-communicating subpopulations. It is likely that combining the two algorithms, perhaps to create mrQSO, would result in a significant reduction in the offline error, taking it to levels well below what either algorithm can achieve individually.

VI. CONCLUSION

In this paper we have presented a new method that can be applied to most if not all Evolutionary Algorithms. We have chosen to test it using SPSO, an algorithm was previously shown to hold promise for improvement of its local convergence speed. By adding the regression to SPSO we were able to significantly outperform the current state of the art algorithm on this problem. It is likely that by combining these algorithms with the regression even better results could be obtained. This is a topic for future research.

The main unknown is the adaptability of this technique to functions that are not smooth or regular on the local scale. It is possible that more complex fitness landscapes may "confuse" the regression, causing it to suggest points well away from the actual optimum. This may be able to be countered by using a more complex function, although overfitting could become a problem. For most common test functions though it is likely the quadratic equation used is more than adequate.

Another promising area of research is in what other ways deterministic heuristics and methods can be combined with EAs to improve performance. Even with functions where steepest descent methods cannot be used, there is still a lot of information that is discarded by the EA. By keeping and analysing this information, it is likely further performance gains can be found.

REFERENCES

[1] J. Branke, "Memory enhanced evolutionary algorithms for changing optimization problems," in *Proceedings of the Congress on Evolutionary Computation*, P. J. Angeline, Z. Michalewicz, M. Schoenauer, X. Yao, and A. Zalzala, Eds., vol. 3. Mayflower Hotel, Washington D.C., USA: IEEE Press, 6-9 1999, pp. 1875–1882. [Online]. Available: citeseer.ist.psu.edu/branke99memory.html

[2] B. Gottfried, *Introduction to Optimization Theory*. Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1973.

[3] M. Clerc and J. Kennedy, "The particle swarm - explosion, stability, and convergence in a multidimensional complex space," in *IEEE Transactions on Evolutionary Computation*, vol. 2, 2002, pp. 58–73.

[4] J. Kennedy and R. Eberhart, *Swarm intelligence*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2001.

[5] X. Li, "Adaptively choosing neighbourhood bests using species in a particle swarm optimizer for multimodal function optimization," in *Proceedings of Genetic and Evolutionary Computation Conference 2004 (GECCO'04) (LNCS 3102)*, 2004, pp. 105–116.

[6] D. Parrott and X. Li, "A particle swarm model for tracking multiple peaks in a dynamic environment using speciation," in *Congress on Evolutionary Computation (CEC2004)*, vol. 1, 2004, pp. 98–103.

[7] S. Bird and X. Li, "Informative performance metrics for dynamic optimisation problems," in *GECCO '07: Proceedings of the 9th annual conference on Genetic and evolutionary computation*. New York, NY, USA: ACM Press, 2007.

[8] S. Bird and X. Li, "Enhancing the robustness of a speciation-based PSO," in *Proceedings of the IEEE International Conference on Evolutionary Computation*, 2006, pp. 843–850.

[9] E. Peer, F. van den Bergh, and A. Engelbrecht, "Using neighbourhoods with the guaranteed convergence PSO," in *Proceedings of the 2003 IEEE Swarm Intelligence Symposium (SIS '03)*, 2003.

[10] T. Blackwell and J. Branke, "Multiswarms, exclusion, and anti-convergence in dynamic environments," *Evolutionary Computation*, vol. 10, no. 4, pp. 459–472, 2006.

[11] X. Li, J. Branke, and T. Blackwell, "Particle swarm with speciation and adaptation in a dynamic environment," in *GECCO '06: Proceedings of the 8th annual conference on Genetic and evolutionary computation*. New York, NY, USA: ACM Press, 2006, pp. 51–58.

[12] F. van den Bergh and A. Engelbrecht, "A new locally convergent particle swarm optimiser," in *Proceedings of the 2002 IEEE International Conference on Systems, Man and Cybernetics*, 2002.