Finding Social Landscapes for PSOs via Kernels

W. B. Langdon and R. Poli

Abstract— **Particle swarm optimiser and genetic algorithm populations are macro-organisms, which perceive their environment as if filtered via a kernel. The kernel assimilates each individual's sensory abilities so that the collective moves using a greedy hill-climbing strategy. This model is fitted to data collected in real PSO and GA runs by using genetic programming to evolve the kernel.**

In nature animals tend to live within groups. The social interactions effectively transform the fitness selection landscape seen by an isolated individual. In some cases a group behaves (or even can be said to think) like a single organism. Kernels provide a lens which coarse-grains or averages individual senses and so may help explain joint actions and social responses.

The original multi-modal problem is smoothed by convolving it with a problem specific filter designed by GP. Because populations see the transformed social fitness landscape, they can pass over local optima. GP can give a good fit between the predicted behaviour of the macroscopic organism and the actual runs.

I. Introduction

In plant and animals systems it is the rule rather than the exception that individuals, at least for some critical periods, live within groups [1]. In a school, individual fish have limited perception, which is dominated by the movement of other members of the school adjacent to them. Also, any food source found by a group member will have to be shared with other members. So, why should individuals stay in a group? A reason is that there are many things that a group of individuals or cells can do that an isolated individual or cell cannot. These increase the survival capabilities of each individual. For example, the group has increased action and reaction capabilities. Some groups are able to: gather rain drops, overshadow competitors, or even build cathedrals. Also, a group may have increased sensing capabilities. For example, fish in a school can rapidly spread information by copying each other's behaviour. This may reveal the presence of food or danger to all members of the group even if only a small number (in fact, as low as 2) of individuals have first hand information. Through these extended sensing and acting capabilities, in many circumstances, a group behaves like a single organism.

One might imagine that in order to model computationally or mathematically these rich sets of behaviours, one would have to use very complex models. This, however, is not necessarily the case. For example, it is possible to observe the emergence of realistic fish school behaviours in models where each fish is a simple agent controlled by forces which implement its desire to both behave like and stay in proximity of other fish, while at the same time keeping a certain distance from other fish [1].

WCCI 2006, 6-21 July, Vancouver, IEEE press. Department of Computer Science, University of Essex, CO4 3SQ, UK.

If we now turn our attention to the world of computation, we see that many population-based search algorithms take inspiration from some natural system involving groups of individuals. Particle Swarm Optimisers (PSOs) [2], [3], [4], for example, take direct inspiration from bird flocking or fish shoaling. In a PSO, particles fly on the fitness landscape under the control of two forces representing cognition and sociality, respectively. The cognition force drives individuals to search around their personal best, sociality drives them to conform their search to that of the best particle in either their neighbourhood or in the whole swarm. Genetic algorithms, GAs, have a different source of inspiration. Nonetheless, they too simulate groups of individuals with some sort of social interaction. Indeed, under the effects of crossover and selection (but no mutation), in typical GAs the offspring sample the subspace represented by the convex hull of the population. So, there is a kind of "social" bias towards sampling areas where other individuals are located. Other population-based algorithms are based on the notion of the search being performed by a population of locally interacting individuals.

As a result of these interactions search algorithms show complex emergent behaviours. For example, the ability to avoid getting stuck in local optima. So too, in these computational systems, groups appear to have, extended sensing and acting capabilities. The question we want to explore in this paper is: *to what degree can we interpret and model the behaviour of a particle swarm optimiser and other artificial population-based optimisation algorithms as that of a corresponding, single macroscopic (or multicellular) individual or swarm?*

In order to answer our main question, we will start by first providing a simplified, but realistic, model of the behaviour of swarms of interacting entities (Section II). The model originally arose and was refined in a series of meetings with biologists (including the authors of [1]) collaborating in a large research project, the Extended Particle Swarms (XPS) project, involving several sites in the UK and numerous international collaborators. Although the model is simple, it explains in which sense the collective can be seen as a single individuals and how such an individual essentially "perceives" its environment as if deformed or filtered via a receptive field or "kernel" which assimilates each individual's senses.

Motivated by this model, the original question can then be recast as whether or not kernels, which allow accurate modelling of particle swarm optimisers and other population-based search algorithms, exist. To help us answer this new question, we decided to apply genetic programming (GP) to explore the space of possible kernels. Our experiments with small PSOs, show that the kernel approach has some credibility. We show several examples where we can model movement of a PSO swarm as a whole, even though PSO swarms are composed of semi-independently moving individuals. In contrast, early work on understanding PSOs, started by considering each individual's movements as if they were isolated from the swarm before looking at their combined effects [5], [6]. In our model, the centre of the swarm responds to the underlying optimisation problem via a receptive field. We model the receptive field by transforming the problem landscape by convolving it with a kernel evolved for this purpose. In principle evolution might have produced any kernel. However, as we shall see, those evolved act as low pass filters, which smooth the landscape [7]. That is, they effectively remove high frequency landscape features, such as local, optima. So, the motion of the population on a landscape can be seen as the motion of a hill-climber on a transformed, smoother landscape. In line with recent research on population dynamics [8], [9], [10], [11] we can see this new landscape as a coarse-grained, population-level version of effective fitness landscape.

The next section gives the mathematical background to the kernel model. Sections III–VI describe and give parameters for the PSO, GA, gradient follower and GP. (Embolden by our success with PSOs, we also evolved kernels to describe more traditional evolutionary algorithms, such as a GA.) The two optimisation benchmarks (3 Gaussians and Rastrigin) are give in Section VII. Finally the implications of the high quality kernels evolved (Section VIII) are discussed in Section IX and our conclusions are given in Section X.

II. Mass-Spring Kernel Model of Emergent Behaviour

The following simple analysis justifies some of the modelling choices we make in the following sections. Let us start by analysing a very simple situation where we assume that each individual is a mass which interacts with other individuals via springs. Let us imagine that each mass has an active control system and can perceive the environment and act accordingly. The only action allowed is to move the mass using an external force.

Let us initially imagine that the springs are extremely stiff, so much so that the population behaves exactly like a rigid body. When an individual decides to generate a propulsion force, all of the population is affected. If more than one individual generates forces, the forces will be added vectorially and the motion of the population will be determined by the resultant force. So, the links between individuals transfer information and coordinate behaviour, without any centralised mechanism.

The situation is not very different if we reduce the stiffness of the springs. As a result of this change it may take longer for information to be integrated along the whole "body" of the population, but eventually information is transferred and, looking at the behaviour over appropriately long time scales, the resulting behaviour is almost identical.

To simplify the analysis, let us imagine that the environment is simply a function $f(x)$ which represents the resource distribution at each point in space. For example, we could imagine that this function represents the density of food. Naturally, we should not expect individuals to be able to perceive the whole environment, so perception will have to be limited to a certain area around each individual. Furthermore, individual animals have to spend time and energy to move to different locations, so it is sometimes preferable to go for a place which is nearer even if it has less resources. Essentially the net value that an animal could gain per unit of food available is some decreasing function $w(d)$ of the distance d from the current location. Naturally, animals prefer places where there are plenty of resources, so the attractiveness of a particular location depends on the average or sum of the available perceived resources. To sum up we can model the *perceived attractiveness* of the environment at any given location x for an individual as

$$a(x) = \int f(y)\omega(x-y)dy$$

That is, *the perceived attractiveness of the environment $a(x)$ is the convolution between the actual food distribution $f(x)$ and a kernel $\omega(x)$ representing the perceptual and locomotion capabilities of the individual.*

We expect each individual to move in order to maximise the perceived attractiveness of its position. This could be modelled by simply computing the gradient of $a(x)$ and moving accordingly (hill climbing). We can, for example, assume that the force generated by each individual would be proportional to $\nabla a(x)$, i.e.

$$F_i = \eta \nabla a(x_i)$$

where η is an appropriate constant, x_i is the position of individual i and $\nabla a(x_i)$ is the gradient of $a(x)$ at x_i. Because of the linearity of convolution we have

$$\nabla a(x) = \int f(y)\nabla\omega(x-y)dy.$$

So, the direction and amplitude of the motion force is proportional to the convolution between the original landscape and the derivative of the receptive field ω.

As mentioned above, each animal is constrained in its motion by its social interactions. Let us start by assuming the interactions are rigid links. Then the motion of the centre of mass $\overline{x}$ of the population is controlled by a force

$$F(\overline{x}) = \sum_i F_i = \eta \sum_i \nabla a(x_i) = \eta \sum_i \nabla a(\overline{x}+\delta_i)$$

where δ_i represents the displacement of individual i from the centre of mass $\overline{x}$, i.e., $\delta_i = x_i - \overline{x}$. So,

$$\begin{aligned} F(\overline{x}) &= \eta \sum_i \int f(y)\nabla\omega(\overline{x}+\delta_i-y)dy \\ &= \eta \int f(y) \sum_i \nabla\omega(\overline{x}+\delta_i-y)dy \\ &= \eta \int f(y)\nabla\omega_p(\overline{x}-y)dy, \end{aligned}$$

where

$$\omega_p(z) = \sum_i \omega(z + \delta_i).$$

In other words, *the motion of the population is controlled by a force which is proportional to the convolution between the resource distribution and the gradient of a new, coarser-grain kernel, ω_p, representing the perceptual and motor capabilities of the population seen as a single individual!*

This macroscopic and emergent animal (swarm) moves in the environment following the gradient of an attractiveness function

$$a_p(x) = \int f(y)\omega_p(x - y)dy.$$

So, the food distribution, which from now on we will interpret as a fitness function, is seen by the population through the looking glass lens of the filter/kernel. Note that the model does not break down even in the special case in which the receptive fields, ω, of each individual are Dirac delta functions, that is when each individual has no knowledge about its neighbourhood. Indeed, in this particular case, ω_p would be a moving-average-type of (low-pass) filter.

III. Particle Swarm Optimiser

In our experiments we use a simple five-member PSO swarm. There is no constriction, instead we use a standard [12, page 1227] PSO where the speed is limited to 1.0. Constriction is often beneficial, e.g. [13], however it can lead to swarms getting stuck at non-optimal values, even at values below nearby local peaks [14]. It is common to use "small" swarms. For example, [15] and [16] both achieved substantial results on real-world problems with only ten particles. However multi-objective problems are often solved with larger swarms [17]. Also genetic search, next section, even for one objective, seems to require bigger populations [18]. This may explain why sometimes [19] PSOs can be more effective than GAs.

To ensure there are some interesting dynamics the swarm is started well away from the optimum. Individuals are given random initial positions uniformly chosen from the range 9.0 to 11.0. Similarly the initial velocities are randomly chosen from their legal range: $-1.0 \ldots +1.0$. Since the natural period of the oscillations of this PSO is 6 [5, Table I] [6, page 1942], the PSO is run for 25 cycles. This allows the first five cycles to be treated as settling time and discarded and still require the model (effectively a hill climber, see below) to match three oscillations. I.e., the GP tries to make the hill climber match the last 20 generations of the PSO run.

IV. Genetic Algorithm

Having applied the kernel approach to swarm systems, we wondered if it could also work with more traditional optimisation techniques. We chose to try it on a genetic algorithm. Again to see interesting dynamics, we start the initial population well away from the optimum and, as the PSO, uniformly randomise the initial chromosomes in the range 9 to 11. Since the GA has no concept of momentum, it is not necessary to allow it five generations to settle. Instead the hill climber is required to match the centre of the GA population immediately. I.e. from generation 0 to 19.

TABLE I. TinyGP Swarm and GA Kernel Parameters

Function set:	$+ - \times$ DIV[a]
Terminal set:	110 terminals, including: The remaining terminals are constants uniformly randomly chosen in the range $0 \ldots 1$
Fitness:	Sum of squared prediction error. See Section VI-A
Selection:	steady state binary tournaments for both parent selection and who to remove from the population
Initial pop:	Trees randomly grown with max depth of 4 (root=0)
Parameters:	Population 10 or 1000. 10% crossover, 90% mutation (2% chance of mutation per tree node).
Termination:	generation 100

[a] If $|y| <= 0.001$ DIV$(x, y) = x$ else DIV$(x, y) = x/y$.

As we are dealing with continuous problems, we used the standard binary-reflected Gray code [20, page 297], [21, page 100] from [22] to encode the range -11 to $+11$.

The GA uses binary selection tournaments with point mutation and one-point crossover on a steady state population [23], [24]. The chromosome length (16 bits), crossover rate (60%), mutation bit flip rate (1/16) and population size (100) are as [18]'s Rastrigin experiments.

V. Hill Climber

The hill climber is started at the centre of mass of the population. Its next twenty moves are compared with the centre of the swarm at successive generations. (In the GA runs, we calculate the mean position of the population every 100 GA fitness evaluations). The memoryless hill climber responds directly to the gradient, accepting all moves, regardless of their fitness. By gradient, of course, we mean the gradient of the landscape resulting from the convolution of the actual landscape with kernel evolved by GP. For numerical stability, the function evolved by GP is treated as the gradient of the kernel, rather than the actual kernel. We do not need the actual kernel during the run. When this is needed (as in Figures 5 and 9) it is reconstructed by integrating its differential after the run.

VI. Evolving Kernels with GP

We use the TinyGP genetic programming system, cf. Table I. It evolves a kernel, i.e. a function, which is convolved with the one-dimensional problem landscape. This gives another landscape which is "perceived" by a hill climber. In these initial experiments, the hill climber is deliberately simple (e.g. it has a fixed learning rate, we chose $\eta = 1$). More sophisticated gradient based techniques are described in [25, pages 276–279]. It updates its position x in the landscape using the rule: $x_{t+1} = x_t + \nabla x$. Where ∇x is the gradient at x. The goal for the GP is to evolve a kernel which causes the hill climber to move so as to resemble movement of the whole PSO swarm.

A. GP Fitness function

The optimiser (i.e. PSO or GA) is run 5 times with independently chosen random initialisations. The fitness of each evolved kernel is the sum, over the last twenty generations

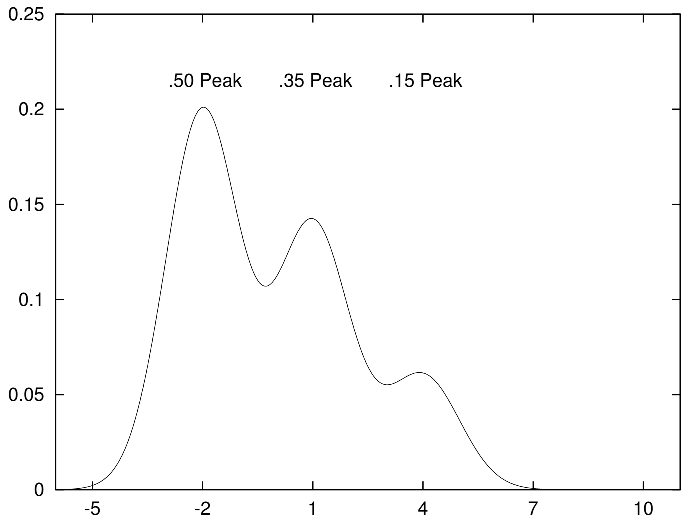

Fig. 1. Sum of three unit variance Gaussians centred at -2, 1 and 4. $0.50N(x,-2,1) + 0.35N(x,1,1) + 0.15N(x,4,1)$

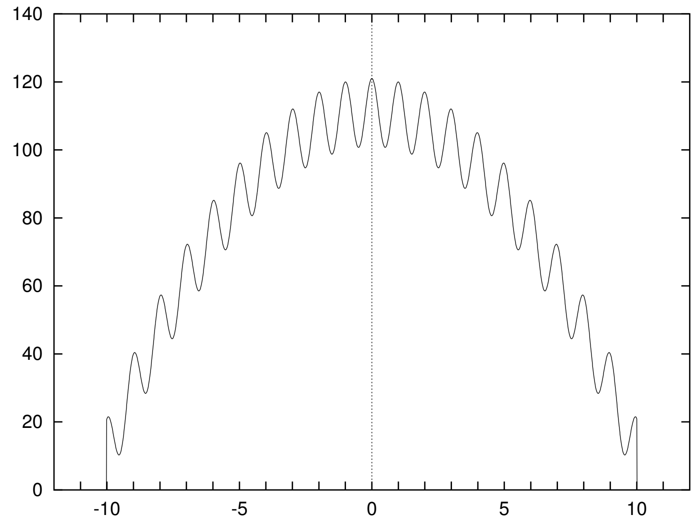

Fig. 2. Rastrigin.

of each run, of the squared distance between the position of the hill climber and the mean position of the members of the population.

To reduce the impact of lucky individuals (who got a high score once due to fortunate random initialisations), the fitness of each potential parent is re-evaluated in each tournament. (The most recent fitness value is used in tournaments to decide who to remove from the population.) In Section VIII (esp. VIII-E) we will see that this gives kernels with low training errors, which also predict the path of optimisers starting from initial conditions never used by the GP. I.e. it helps to avoid overfitting.

VII. Demonstration Landscapes

A. 3 Gaussians

The first test landscape is asymmetric and multi-modal. It is created by adding together three different sized Gaussian curves, each with unit variance, centred at -2, 1 and 4, cf. Figure 1.

B. Rastrigin

The Rastrigin benchmark is widely used since its many local peaks give interesting dynamics [25], [12, page 149]. As suggested in [26], we easily converted it to a maximisation problem. We clip it so that it is only non-zero inside the range -10 to $+10$. See Figure 2.

VIII. Results

A. PSO on 3 Gaussians

The behaviour of the five particle PSO and the hill climber on the kernel (evolved in the first GP run with a population of ten trees) are given in Figure 3.

In all runs the swarms are able to climb upto the first peak but are not trapped by it. Instead they all get at least as far as the second peak. Here in some runs the swarms diverge; some succeed in passing over it and start oscillating about the

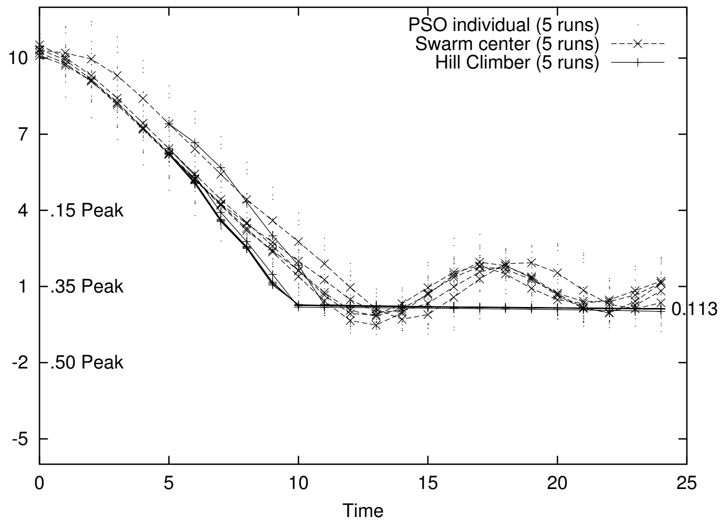

Fig. 3. Testing evolved kernel. Comparison of five actual PSO runs (dashed lines) with five hill climbing runs on three Gaussians landscape (solid lines) Training RMS error 0.91 v. test RMS error 0.87

highest fitness value. However in all of the first five test runs (Figure 3) the swarm oscillates about the intermediate peak. Note this divergence does not prevent GP transforming the landscape so that the hill climber usually follows the centre of the swarm. The average difference between the hill climber and the mean position of the swarm at each of the twenty PSO update cycles (i.e. 5–24) is given in Table II.

As with the other kernels, GP uses the protected division operator to evolve a sharp spike in the differential of the kernel. The evolved function is plotted in Figure 4, whilst Figure 5 shows the kernel. (The kernels are obtained by numerical integration of their differentials). While Figure 6 shows the gradient (solid line) produced by convolution and the corresponding effective fitness landscape (dashed line).

TABLE II. Performance of Evolved Differential Kernels (5 runs)

	GP Pop	RMS Train error	RMS Test error
PSO 3 Gaussians	10	0.91	0.87
PSO Rastrigin	1000	0.47	0.58
GA 3 Gaussians	10	0.91	1.02
GA Rastrigin	1000	0.45	0.65

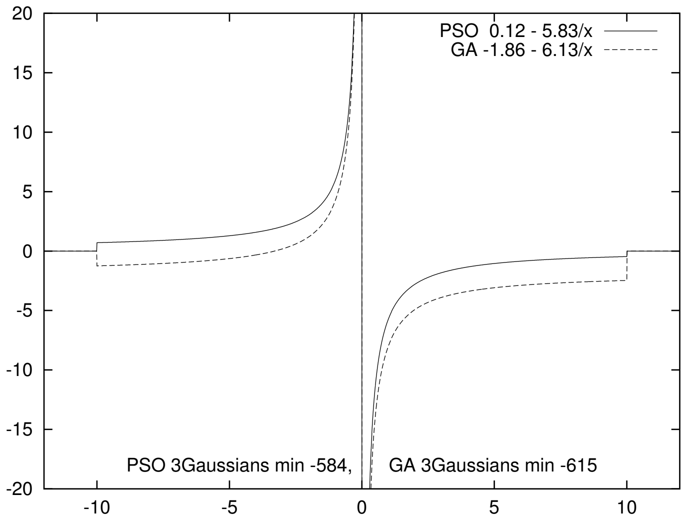

Fig. 4. Differential of kernels for 3 Gaussians problem. (evolved in first GP run, pop 10). Convolutions are truncated to zero outside range $-10 \ldots +10$.

B. PSO on Rastrigin

The five members of the PSO easily step over the multiple local optima traps in the Rastrigin problem and oscillate near the global optimum. The first GP run, with a population of 1000 trees, evolved a kernel which transforms the landscape seen by the five members of the PSO swarm into a landscape where a simple hill climber emulates their centre of mass. See Figures 7, 8, 9 and 10.

C. GA on 3 Gaussians

Figure 11 shows paths taken by the hill climber on the transformed landscape produced by the first GP run (10 trees). Notice that the hill climber takes almost the same path in each of the five test runs. This is not a surprise since it is started from very similar positions (the mean of the GA population, of 100 bit strings) each time. In fact, the hill climber paths tends to converge over time. Figure 11 also shows the location of every individual in each of the five GA populations over time. (Noise along the x-axis has been added in Figure 11 to spread the data.) We can see, despite the high point mutation rate, in each run the population clusters tightly near the optimum. The gradient in the transformed landscape causes the hill climber to oscillate nearby. (Actually between -3 and -2.)

The evolved function is plotted in Figure 4 and the corresponding kernel is in Figure 5. (GA, dashed lines in both graphs). The result of convolving with the three Gaussians is given in Figure 12.

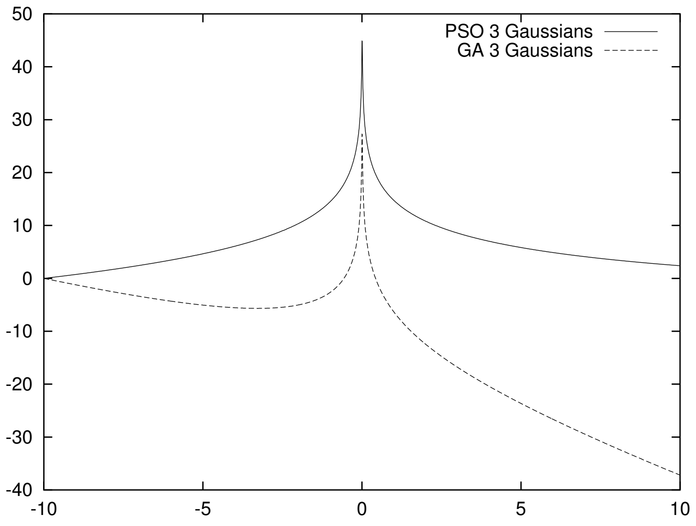

Fig. 5. Kernels evolved in first GP run for 3 Gaussians (pop 10) (Obtained by numerical integration of curves in Figure 4)

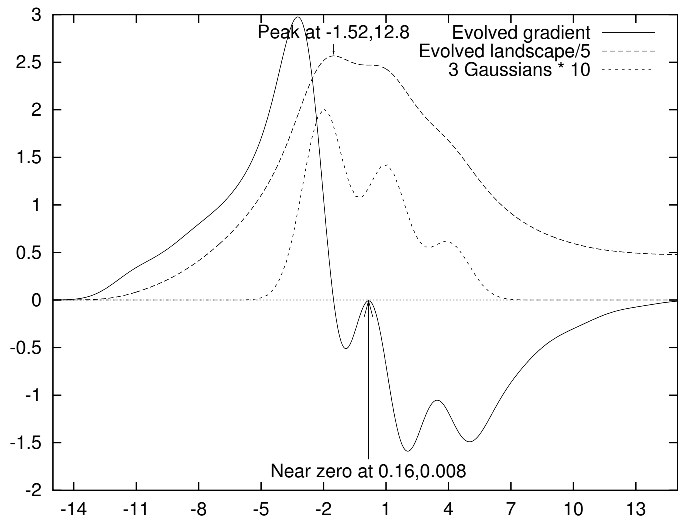

Fig. 6. Gradient used by hill climber for PSO 3G. Gradient calculated by convolving 3 Gaussians with evolved differential kernel (cf. Figure 4, PSO). For comparison (linearly rescaled) 3 Gaussians and new landscape (obtained by integration) are also plotted.

D. GA on Rastrigin

The first GP run, with a population of 1000 trees, evolved a kernel (Figures 8 and 9, dashed line) which transforms the Rastrigin landscape so that the hill climber tracks the mean of the GA population. (Again our simple hill climber oscillates a little.) The GA populations move rapidly towards the optimum (at the origin) easily stepping over the multiple local optima traps. The first five test runs are shown in Figure 13. (As before, in Figure 13, noise is added to spread out horizontally the individual members of the five populations.) Figure 14 gives the gradient (solid line) of the transformation of Rastrigin, produced by convolution with the evolved function (cf. Figure 8, GA). Notice again the effective landscape (dashed line) has been smoothed with

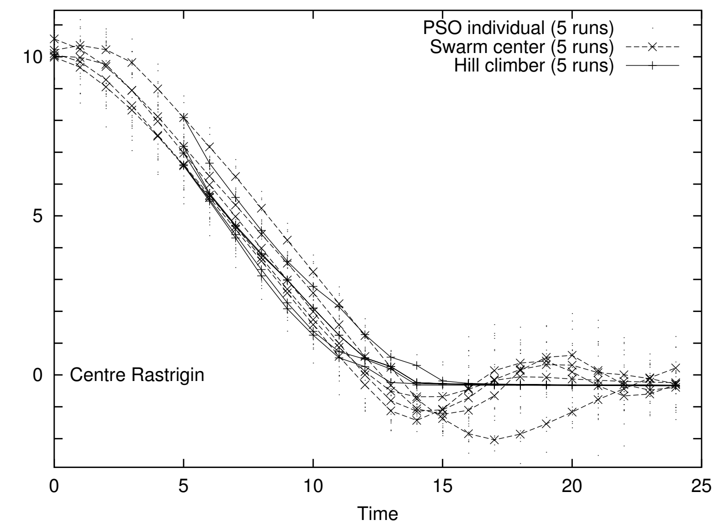

Fig. 7. Comparison of first five PSO test runs on Rastrigin with five hill climbing runs (solid lines) on kernel transformed runs Training RMS error 0.47 v. test error 0.58.

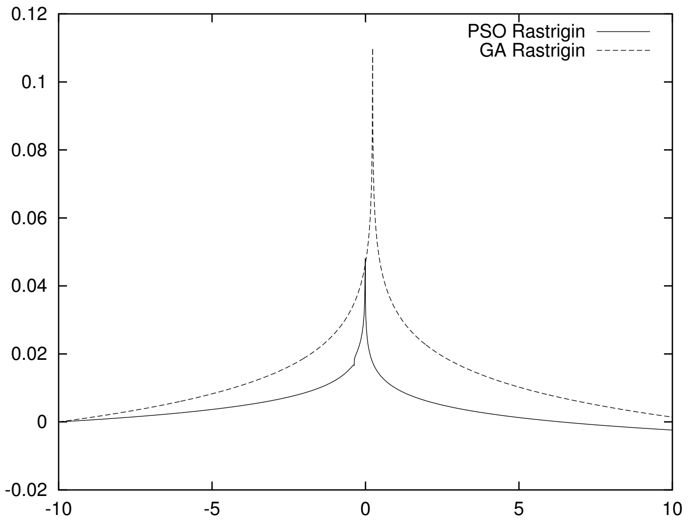

Fig. 9. Kernels evolved in first GP run for Rastrigin (pop 1000). (Obtained by numerical integration of curves in Figure 8)

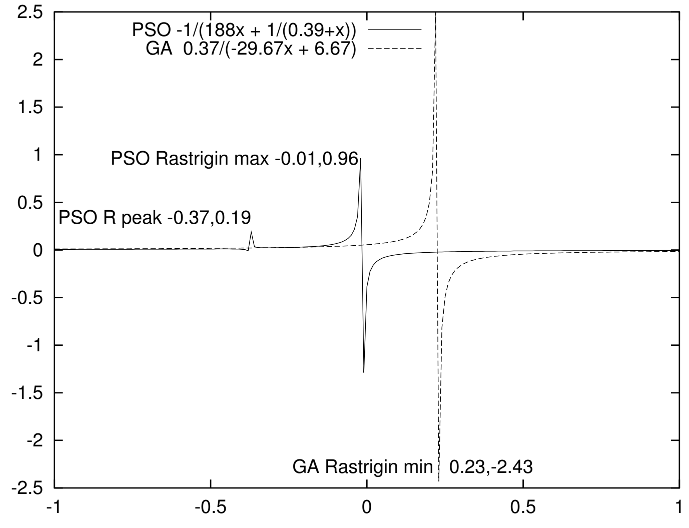

Fig. 8. Differential of kernels for Rastrigin (evolved in first GP run, pop 1000).

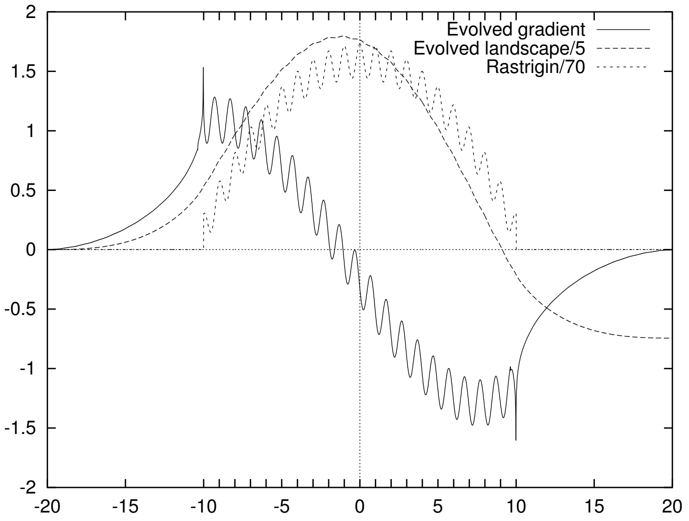

Fig. 10. Gradient used by hill climber for PSO Rastrigin. Gradient calculated by convolving Rastrigin with evolved differential kernel (cf. Figure 8, PSO). For comparison (linearly rescaled) Rastrigin and new landscape (obtained by integration) are also plotted. Note the transformed landscape is smooth but its optimum is slightly offset wrt. the original.

respect to the original Rastrigin (dotted line).

The high mutation rate means that, while most of the population clusters near the origin, a small number of individuals are well away from it. These extreme members of the population have a proportionately large impact on the mean so that it may be slightly (≈ 0.2) away from the origin.

E. Generality of Evolved Differential Kernels

As Tables II and III make clear, performance when tested with different random initial conditions, is not significantly different from that when each differential kernel was trained. That is, the kernels do not over fit their training data. However they have adapted to their environment, in that they are useless when faced with a different problem.

Extremely high errors are seen for both differential kernels trained on the three Gaussians landscape when faced with Rastrigin (Table III). These are due to the first step of the hill climber. The gradient of the transformed landscape is in the direction of the global optimum but is very steep. I.e. the scaling constants appropriate to the three Gaussians problem are not appropriate for Rastrigin. (Note the difference in vertical scales of Figure 1 v. 2 and Figure 4 v. 8.) The gradient is so big that the hill climber is directed to make an enormous leap. This takes it far out of the true range of the problem, where there is no gradient. Therefore the hill climber remains stuck at a large negative value, which, in turn, gives the huge RMS error.

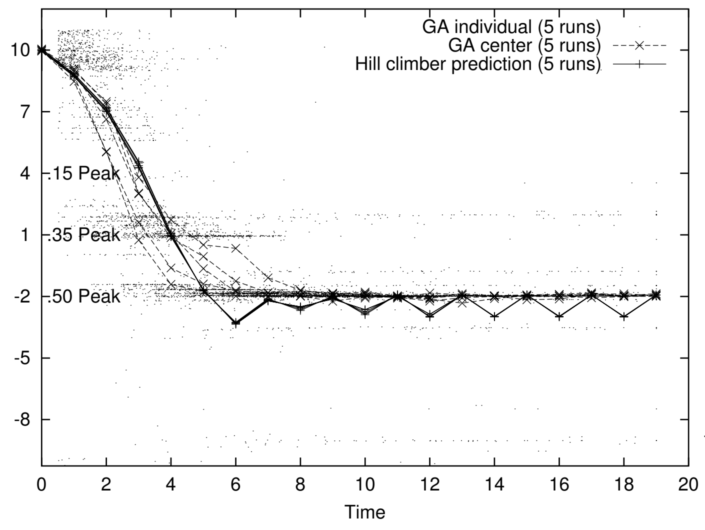

Fig. 11. Comparison of first five GA test runs on three Gaussians problem with five hill climbing runs (solid lines) on kernel transformed landscape. Training RMS error 0.91 v. test error 1.02.

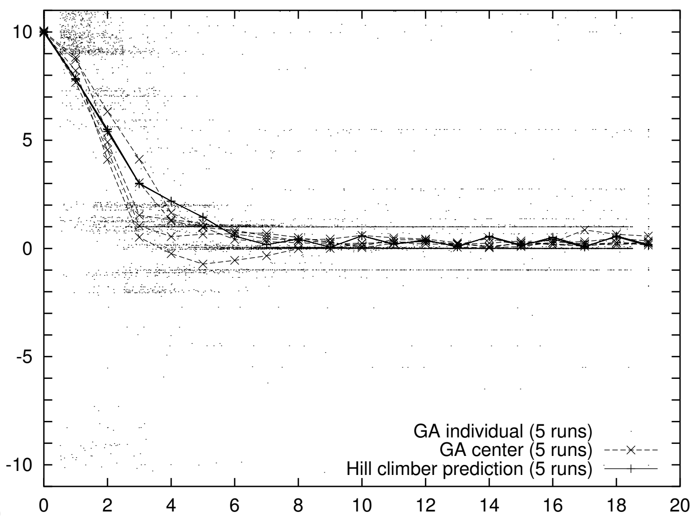

Fig. 13. Comparison of first five GA test runs on Rastrigin with five hill climbing runs (solid lines, GP pop 1000) on kernel transformed landscape. Training RMS error 0.45 v. test error 0.65.

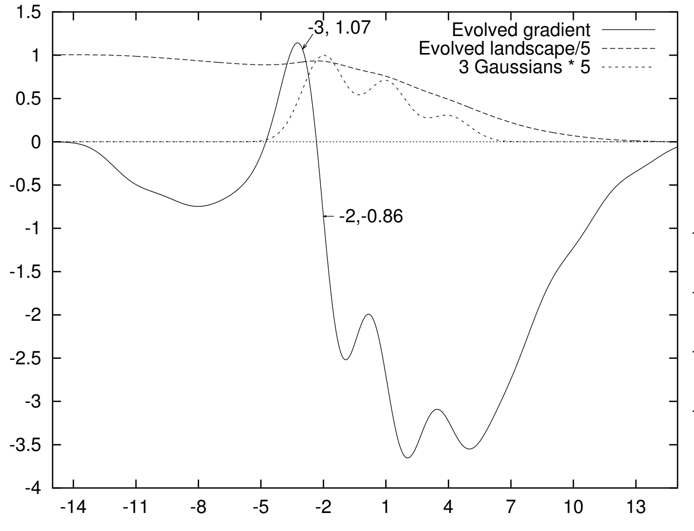

Fig. 12. Gradient used by hill climber for GA 3G. Gradient calculated by convolving 3 Gaussians with evolved differential kernel (cf. Figure 4, GA). For comparison (linearly rescaled) 3 Gaussians and new landscape (obtained by integration) are also plotted.

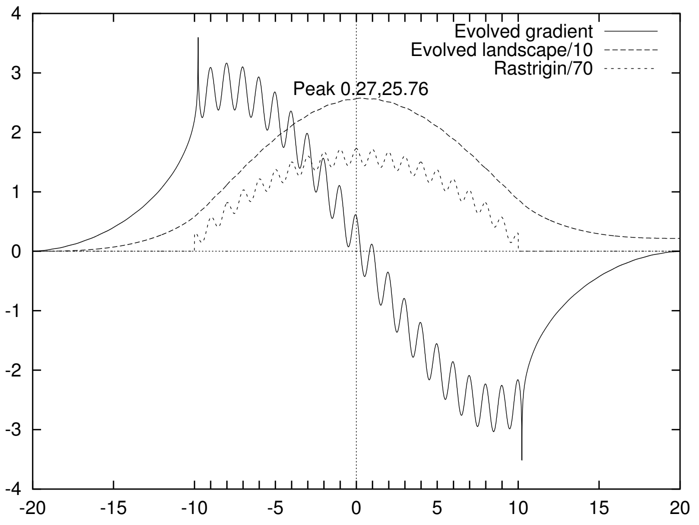

Fig. 14. Gradient used by hill climber for GA Rastrigin. Gradient calculated by convolving Rastrigin with evolved differential kernel (cf. Figure 8, GA). For comparison (linearly rescaled) Rastrigin and new landscape (obtained by integration) are also plotted.

TABLE III. Test RMS error of evolved differential kernels on both problems (means of 25 runs). Note: differential kernels are specific to the environment where they were trained and in all 4 cases perform much worse in a different environment.

	GP Pop	3 Gaussians		Rastrigin	
		mean	(sd)	mean	(sd)
PSO 3 Gaussians	10	0.97	(0.21)	1020.00	(132.00)
PSO Rastrigin	1000	5.30	(0.69)	0.66	(0.29)
GA 3 Gaussians	10	1.10	(0.30)	2700.00	(132.00)
GA Rastrigin	1000	10.64	(0.29)	0.86	(0.35)

IX. Discussion

These experiments actually provide the effective landscape seen by simple particle swarm optimisers and evolutionary algorithms. It is nice that, as so often assumed, those found by genetic programming are "smoothed" versions of the true fitness landscape. We think this result is general, and, in future research, we will seek experimental confirmation in real-world problems. Naturally, given their typically enormous size and the costs associated with convolutions, we expect to be able to do so only where kernels are small and limited resolution is needed.

Also, we believe that it may be possible to apply our kernel-based approach to natural swarms, in particular fish shoal and flocks of chickens, as well as in the time domain. That is, in the same way that a spatial kernel can model a group of individuals spread over some area of space, a temporal kernel can show the effects of an individual remembering sensory readings from its past locations. I.e. one might try to use kernel methods to show the benefits and drawbacks of memory. One could even think of combined space and time kernels, showing how a collective uses "group memory".

Naturally, we expect that there are situations where it may be impossible to find a single kernel which describes the behaviour of the swarm as a whole. For example, it is possible that modern explorative PSOs [12] are too dynamic for the swarm to behave as a cohesive whole.

So far we have looked at static landscapes but natural environments, and indeed many engineering challenges, are dynamic and multiple threats must be overcome simultaneously. For dynamic and multi-objective problems, PSOs can do better than other approaches [27], [28], [17]. Of course real life is much more complicated and multiple interacting problems must be solved. Co-evolution [29] suggests an approach. It is possible, both in coevolution and where there is a need to find all Pareto undominated solutions, that a swarm will lose cohesion and so predicting its average behaviour would be impossible or not useful. However it would be interesting to apply our kernel approach to the time dimension and see if it reveals how such multi-swarm PSOs filter problems to give smooth dynamic fitness landscapes.

X. Conclusions

We have shown on two problems that genetic programming can automatically find simple kernels which transform the optimisation problem seen by individual members of a PSO swarm into a social landscape on which a single-point hill-climber matches the average behaviour of the whole swarm. Given the stochastic nature of PSOs, the agreement between the movement of the single point and the PSO centre of mass is surprisingly good. It is also pleasing, given their different behaviours [30, page 270], that the same approach can also be used with genetic algorithm populations.

The no free lunch theorems and considerable experimental work [31] confirm that there is no universal problem solver. Instead we must continue to analyse both problems and solvers, in the hope of finding better matches between them. By showing, in principle, we can generate coarse-grained effective landscapes on which simple hill climbers move in step with sophisticated algorithms, such as PSOs, we have strengthened the fitness landscapes metaphor and contributed an important analysis tool.

Acknowledgement

We would like to thank Darren Croft, Iain Couzin and Chris Stephens. Supported by EPSRC XPS grant GR/T11234/01.

References

[1] I. D. Couzin, J. Krause, N. R. Franks, and S. A. Levin, "Effective leadership and decision-making in animal groups on the move," *Nature*, vol. 433, pp. 513–516, 3 February 2005.

[2] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *Proc. 1995 IEEE Int. Conf. on Neural Networks*. pp. 1942–1948.

[3] ——, *Swarm Intelligence*. Morgan Kaufmann, 2001.

[4] A. P. Engelbrecht, *Fundamentals of Computational Swarm Intelligence*. Wiley, November 2005.

[5] M. Clerc and J. Kennedy, "The particle swarm - explosion, stability, and convergence in a multidimensional complex space," *IEEE Trans. on Evolutionary Computation*, vol. 6, no. 1, pp. 58–73, 2002.

[6] E. Ozcan and C. K. Mohan, "Particle swarm optimization: Surfing the waves," in *CEC-99*. pp. 1939–1944.

[7] A. v. E. Conradie, R. Mikkulainen, and C. Aldrich, "Intelligent process control utilising symbiotic memetic neuro-evolution," in *CEC2002*, IEEE Press, 2002, pp. 623–628.

[8] C. Stephens, ""effective" fitness landscapes for evolutionary systems," in *CEC-99*. Washington D.C., USA, 6-9 July 1999, pp. 703–714.

[9] C. R. Stephens and J. M. Vargas, "Effective fitness as an alternative paradigm for evolutionary computation I: General formalism," *Genetic Programming and Evolvable Machines*, vol. 1, no. 4, 363–378, 2000.

[10] ——, "Effective fitness as an alternative paradigm for evolutionary computation II: Examples and applications," *Genetic Programming and Evolvable Machines*, vol. 2, no. 1, pp. 7–32, Mar. 2001.

[11] P. F. Stadler and C. R. Stephens, "Landscapes and effective fitness," *Comments on Theoretical Biology*, vol. 8, no. 4-5, pp. 389–431, 2003

[12] R. A. Krohling, "Gaussian particle swarm with jumps," in *CEC-2002*. Edinburgh, UK: IEEE Press, 2-5 September 2005, pp. 1226–1231.

[13] M. P. Wachowiak, R. Smolikova, Y. Zheng, J. M. Zurada, and A. S. Elmaghraby, "An approach to multimodal biomedical image registration utilizing particle swarm optimization," *IEEE Transaction on Evolutionary Computation*, vol. 8, no. 3, pp. 289–301, June 2004.

[14] W. B. Langdon and R. Poli, "Evolving problems to learn about particle swarm and other optimisers," in *CEC-2005*. pp. 81–88.

[15] A. Conradie, R. Miikkulainen, and C. Aldrich, "Adaptive control utilising neural swarming," in *GECCO 2002*. pp. 60–67.

[16] N. Jin and Y. Rahmat-Samii, "Parallel particle swarm optimization and finite-difference time-domain (PSO/FDTD) algorithm for multiband and wide-band patch antenna designs," *IEEE Transactions on Antennas and Propagation*, vol. 53, no. 11, pp. 3459–3468, November 2005.

[17] X. Hu and R. C. Eberhart, "Multiobjective optimization using dynamic neighborhood particle swarm optimization," in *CEC2002*, Honolulu, Hawaii, USA: IEEE Press, 2002, pp. 1677–1681.

[18] M. A. Potter and K. A. De Jong, "A cooperative coevolutionary approach to function optimization," in *PPSN III*. 1994, pp. 249–257.

[19] R. J. W. Hodgson, "Partical swarm optimization applied to the atomic cluster optimization problem," in *GECCO 2002*. pp. 68–73.

[20] D. Whitley, L. Barbulescu, and J.-P. Watson, "Local search and high precision gray codes: Convergence results and neighborhoods," in *FOGA 6*, W. N. Martin and W. M. Spears, Eds. 2001, pp. 295–311.

[21] D. E. Goldberg, *Genetic Algorithms in Search Optimization and Machine Learning*. Addison-Wesley, 1989.

[22] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C*. Cambridge University Press, 1992.

[23] G. Syswerda, "Uniform crossover in genetic algorithms," in *ICGA-89*, J. D. Schaffer, Ed. Morgan Kaufmann, 1989, pp. 2–9.

[24] ——, "A study of reproduction in generational and steady state genetic algorithms," in *FOGA*, G. J. E. Rawlings, Ed. pp. 94–101, 1991.

[25] F. van den Bergh, "An analysis of particle swarm optimizers," Ph.D. dissertation, Dept. of Computer Science, University of Pretoria, 2001

[26] X. Li, "Adaptively choosing neighbourhood bests using species in a particle swarm optimizer for multimodal function optimization," in *GECCO-2004*, K. Deb, *et al.*, Eds. Springer-Verlag, pp. 105–116.

[27] T. M. Blackwell and P. J. Bentley, "Dynamic search with charged swarms," in *GECCO 2002*, W. B. Langdon *et al.*, Eds. pp. 19–26.

[28] T. Blackwell and J. Branke, "Multi-swarms, exclusion, and anti-convergence in dynamic environments," *IEEE Trans. EC*, forthcoming.

[29] R. A. Krohling, F. Hoffmann, and L. dos Santos Coelho, "Co-evolutionary particle swarm optimization for min-max problems using gaussian distribution," in *CEC-2004*. pp. 959–964.

[30] G. Dozier, D. Brown, J. Hurley, and K. Cain, "Vulnerability analysis of immunity-based intrusion detection systems using evolutionary hackers," in *GECCO-2004*, K. Deb *et al.*, Eds., pp. 263–274.

[31] T. Bartz-Beielstein, K. E. Parsopoulos, and M. N. Vrahatis, "Design and analysis of optimization algorithms using computational statistics," *Appl. Num. Anal. Comp. Math*, vol. 1, no. 2, pp. 413–433, 2004.