# Multi-Modal Optimisation using a Localised Surrogates Assisted Evolutionary Algorithm

Jonathan E. Fieldsend
Computer Science
University of Exeter
Exeter, EX4 4QF
Email: J.E.Fieldsend@exeter.ac.uk

***Abstract*—There has been a steady growth in interest in niching approaches within the evolutionary computation community, as an increasing number of real world problems are discovered that exhibit multi-modality of varying degrees of intensity (modes). It is often useful to locate and memorise the modes encountered – this is because the optimal decision parameter combinations discovered may not be feasible when moving from a mathematical model emulating the real problem to engineering an actual solution, or the model may be in error in some regions. As such a range of disparate modal solutions is of practical use. This paper investigates the use of a collection of localised surrogate models for niche/mode discovery, and analyses the performance of a novel evolutionary algorithm (EA) which embeds these surrogates into its search process. Results obtained are compared to the published performance of state-of-the-art evolutionary algorithms developed for multi-modal problems. We find that using a collection of localised surrogates not only makes the problem tractable from a model-fitting viewpoint, it also produces competitive results with other EA approaches.**

## I. INTRODUCTION

Optimisation problems in the real world often exhibit a degree of multi-modality, be it only a few modes to contend with, or potentially a vast number. That is, in a particular volume of design space there may be more than one solution which performs equally as well as another, but the regions *between* these solutions map to quality values distinctly *less* good. These are often conceptualised as *peaks* (or *troughs* if the quality measure is to be minimised) – and are referred to as niches or modes.

Decision makers are often interested in locating disparate peaks, as they can offer insight into the behaviour of the problem they are dealing with. Additionally, by discovering parameter combinations which have the equivalent (or similar) behaviour, but which are distributed widely in design space, a wide range of good potential design solutions can be subsequently assessed. Often the model optimised is a software emulation of a physical process, whose mapping may not be exact in all regions, therefore a spread of good solutions is typically a desired output; as realising a particular parameter combination may turn out to be infeasible from a manufacturing point of view, or when manufactured may not behave as emulated.

Many methods exist to search for multiple optima, with fitness-sharing and crowding [1] being two popular evolutionary computation (EC) approaches. Essentially these promote regional subpopulations of a search population, who are concerned with optimising separate modes. As recently highlighted in [2], these algorithms are often highly parameterised themselves, and rely on well-chosen values to perform as required. Other approaches that have been developed for use within evolutionary algorithms (EAs) include clustering [3], derating [4], restricted tournament selection [5], speciation [6], and stretching and deflation [7] (to name but a few).

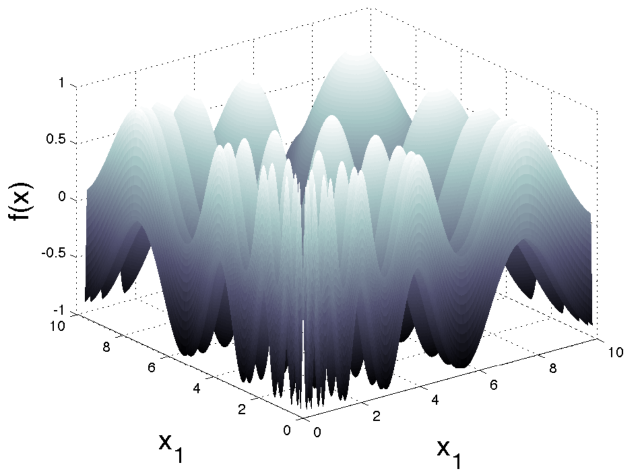
Here we take a different approach to niching, fitting local surrogate models to regions of the design space, and select new design parameters to assess based on the prediction of these models. The number of local modes is not predefined, allowing the algorithm to *learn* the degree of multi-modality as it searches the design space. It accomplishes this by merging regions covered by surrogates, and by searching in new regions via both recombination and speculatively looking in new areas. Results for this localised surrogates assisted evolutionary algorithm are presented on a number of multi-modal test functions from the literature [8]–[10].

The paper proceeds as follows. In Section II we provide a short description of the general multi-modal optimisation problem, this is followed by a short overview of surrogate-based optimisation for multi-modal problems and local model fitting in Section III. In Section IV the multi-modal optimisation algorithm is described, and in Section V its empirical performance is compared to that of a number of recently developed algorithms on a range of standard test problems. The paper concludes with a discussion in Section VI.

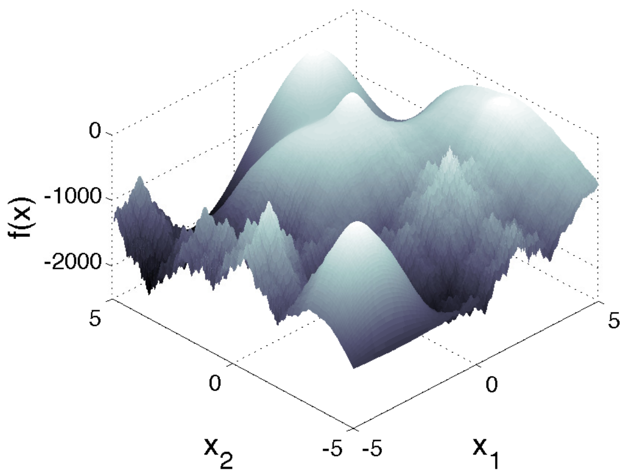
## II. MULTI-MODAL LANDSCAPES

Many methods have been developed for locating multiple optima. From the basic 'Multistart' approach, which applies local search from randomly generated locations [11] – to more sophisticated techniques like fitness sharing and crowding, which both fall in the broader area of *niching* methods. Holland [12] first introduced the fitness sharing concept, which was later refined as a means to partition the search population into different subpopulations based on their fitness [13]. A succinct overview of these general ideas is presented in [1].

The general aim in multi-modal optimisation is similar to that of global optimisation, that is, without loss of generality, we seek to maximise $f(\mathbf{x})$, where $\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^K$ – the feasible domain as defined by any equality and inequality constraints. In the case of a multi-modal problem, we seek not simply to discover a single $\mathbf{x} \in \mathcal{X}$ which maximises $f(\mathbf{x})$, but the set $\{\mathbf{x}^*\} \in \mathcal{X}$ of solutions which obtain the maximum



(a) Inverted Vincent function.



(b) Composite function 1.

Fig. 1. Example multi-modal function landscapes: (a) has 36 global optima, (b) has a mixture of six global modes and many local modes.

possible function response, but which inhabit isolated peak regions. That is, the mapped objective values in the immediate region of an $\mathbf{x}^*$ are all lower than $f(\mathbf{x}^*)$. Two example multi-modal problems are shown in Figure 1. Figure 1a has an asymmetric distribution of many global optima, Figure 1b has fewer global optima, but many *local* optima. Local optima (local modes/peaks) are locations which are surrounded in the immediate vicinity with less '*fit*' solutions (lower responses from $f(\cdot)$), but which do not themselves have the highest possible fitness.

## III. Surrogate-based optimisation

The use of *surrogates* within evolutionary optimisation algorithms has started receiving serious attention in the last decade, a good recent overview of area can be found in [14]. Many industrial problems are expensive to evaluate, and to mitigate this surrogate models fit a estimate of the cost function (based on previously evaluated design parametrisations, and any prior knowledge available). These can then be used in combination with an EA to guide the search process (although reference back to the real fitness function is required in order to update the surrogate – and to compensate for any falsely induced maxima). Most areas of research in surrogates are concerned with the type of surrogate used, and how the surrogate will be integrated within the search process (the 'model management' problem [14]).

Some previous studies have used surrogate approaches for multi-modal problems – in [15] a surrogate is used to effectively 'smooth' the cost landscape as a means to eradicate local minima for problems with many local modes and a single maxima. In [16] local weighted ensembles of surrogates are used for global optimisation, along with lower order polynomial surrogates to also *smooth* local optima as in [15]. In [11] a global surrogate is used in conjunction with a local evolutionary search (which exploited memory for good search directions [17]) – although they were concerned with finding *a* global peak, rather than all global peaks. Most recently in [18] a classifier was fitted locally to each offspring in a differential evolution algorithm, to predict their relative rank order when solutions previously evaluated in their local neighbourhood had a mixed range of fitnesses.

One drawback of many sophisticated models used as surrogates is that they can often themselves take time to *fit* the data, and often performance (time to learn) scales poorly with the number of data points. Where the landscape is highly muti-modal, it is often highly flexible and parameterised surrogates that are required. The use of these becomes rapidly intractable as the number of data samples rises, as, unless the problem being optimised is extremely expensive, the time-cost of regularly fitting the surrogate soon outweighs the time cost of evaluating the actual objective function for medium to long runs. We attempt to side-step this issue here by using a *collection* of local surrogates, whose remit is only a very small region in $\mathcal{X}$, centred on a particular niche estimate, and are therefore very cheap to fit in comparison to a global model. This approach is also useful as the maxima from a globally fitted surrogate may not be as accurate as that induced from a model fitted to just those points sampled in the location of the globally estimated maxima (see e.g. [19]). In [20] it is also observed that the global error of a surrogate may not be a good indication of its *local* error - and as such local fidelity tracking is used to decide when a global model should be refitted to new data.

Here we use the "Design and Analysis of Computer Experiments" (DACE) approach as our local surrogate model [21], specifically a zero order polynomial, with Gaussian correlation model from the Matlab Kriging toolbox [22]. An example of this model is provided in Figure 2, where $y = f(x)$. Note however this illustration is on a global fit, whereas we will be concerning ourselves with local neighbourhood models. It is worth mentioning at this juncture, that although we use Kriging here (also known as a Gaussian Process [23]), the evolutionary algorithm proposed below could have any collection of surrogates embedded within it.

## IV. Proposed algorithm

Rather than train a single surrogate for the entire cost landscape, here we take the approach of having *many* small

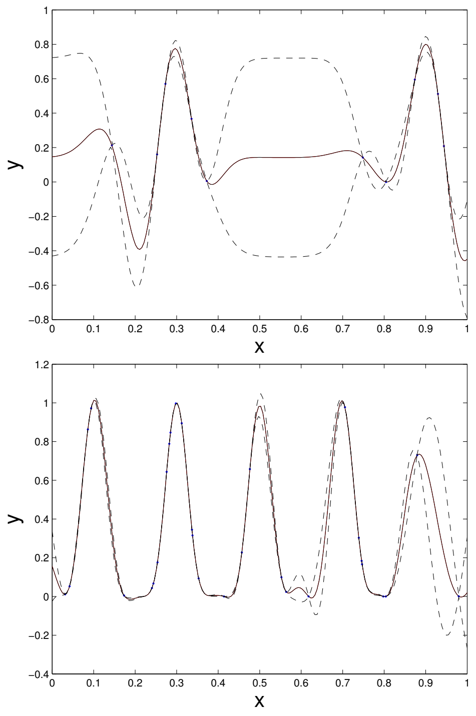


Fig. 2. Kriging surrogate fitness estimate on the equal maxima test function. *Top*: based on 10 random samples of $x$. *Bottom*: based on 30 random samples of $x$. The dashed lines show +/- the root mean squared error estimate provided by the fitted surrogate at each $x$ location.

```
Require: $n, max\_evals$
 1: $\mathbf{X} :=$ latin_hypercube_sample($n$)
 2: $\mathbf{Y} :=$ evaluate($\mathbf{X}$)
 3: $evals := n$
 4: $t := 0$
 5: while $evals < max\_evals$ do
 6:     $\{\mathbf{X}, \mathbf{Y}, evals\} :=$ compare_niches($\mathbf{X}, \mathbf{Y}, evals$)
 7:     $\mathcal{S} :=$ fit_local_surrogates($\mathbf{X}, \mathbf{Y}$)
 8:     $\{\mathbf{X}, \mathbf{Y}, evals\} :=$
        propose_via_surrogates($\mathcal{S}, \mathbf{X}, \mathbf{Y}, evals$)
 9:     $t := t + 1$
10:     if $t = 5$ then
11:         $\{\mathbf{X}, \mathbf{Y}, evals\} :=$
            crossover_niches($\mathbf{X}, \mathbf{Y}, evals$)
12:         $t := 0$
13:     end if
14:     $\{\mathbf{X}, \mathbf{Y}, evals\} :=$
        generate_random_niche($\mathbf{X}, \mathbf{Y}, evals$)
15: end while
16: $\{X^*, Y^*\} :=$ extract_peak_members($\mathbf{X}, \mathbf{Y}$)
17: return $\{X^*, Y^*\}$
```

Fig. 3. High-level pseudocode of the localised surrogates evolutionary algorithm.

surrogates, who are concerned only with the local landscape surrounding the currently estimated niches.

Because we only fit the surrogate to a small volume of $\mathcal{X}$, we only use previously evaluated solutions in the immediate vicinity to fit the model parameters, thus model fitting at each generation is extremely rapid, even though there may be many surrogates fitted. Indeed, it is orders of magnitude quicker than fitting a single model for all the data (as fitting a single model is an $\mathcal{O}(n^3)$ operation – where $n$ is the number of data points used). The algorithm is self-adaptive, with the number of niches maintained depending on the properties of the landscape discovered, and it seeks to balance both exploration of the space for previously undiscovered niches, and exploitation of the niches found thus far. As such, the number of niches in $\mathcal{X}$ does not need to be known *a priori*, or a vast search population maintained right from the start – instead the number of niches dynamically shrinks or expands as the search progresses. The algorithm maintains a set of sets of *niche histories* $\mathbf{X}$, where $\mathbf{X}_i$ is the set of $m$ designs $\{\mathbf{x}\}_{j=1}^{m}$, which are used to define the immediate peak region of the $i$th niche. Each $\mathbf{X}_i$ therefore includes a single $\mathbf{x}^*$ estimate. $\mathbf{Y}_i$ is the collection of corresponding responses from $f(\cdot)$ for the elements of $\mathbf{X}_i$.

The algorithm at a high-level is described in Figure 3. The initial number of random solutions to be evaluated, $n$, is inputted (these form the basis of the initial estimated niches), along with the maximum number of true function evaluations permitted for the algorithm (as distinct from surrogate evaluations). The algorithm proceeds as follows. The initial solutions are drawn from latin hypercube sampling of $\mathcal{X}$ (line 1), which also form the initial niche histories. On line 6 the niches are compared to one another – and those that are concerned with the *same* niche are merged (this is detailed further in Figure 4). After this, local surrogates are fitted to each niche (line 7, and Figure 5), and on line 8 these surrogates are used to evaluate a number of solutions proposed in the vicinity of each niche. One of these is then evaluated on the actual objective function, $f(\cdot)$, (details of proposal selection are provided in Figure 6). In combination with the current niche peak estimates these newly evaluated locations are used to improve the niche estimate.

Every five generations the niche peaks are crossed over (line 11) using simulated binary crossover (SBX) [24]. The resulting children are then placed in new niches (although these may subsequently be merged when the algorithm loops back to line 6).

The `compare_niches` subroutine, as outlined in Figure 4, is concerned with *reducing* the number of niches maintained, as two niches may be climbing up the same peak from different directions (and therefore are concerned in actuality with the *same* niche). First all niches whose peaks have changed since the last generation are marked (i.e. those that are brand new, or have had a higher point in the niche locality found in the previous iteration). The closest *other* niche peak (in design space) to each of these marked niches is also found. Once all pairings have been calculated the procedure then processes each pair in turn[1] (lines 4-17). If the two peaks are within some small tolerance of each other in design space (here set at $1.0 \times 10^{-3}$), then the niches are automatically merged (lines 5-6), otherwise the location midway between the two peaks is evaluated (lines 8-10). The midpoint, $\mathbf{x}'$, is then compared to its two parents. If it is worse than both parents then the niches are maintained separately and the evaluated midpoint is simply added to the niche histories of both peaks (as it may

[1]Note, to avoid wasted computation it is best to check the list produced, as two changed points may be the closest to each other, thus duplicating pairs.

```
Require: $\mathbf{X}, \mathbf{Y}, evals$
 1: $\{X^*, Y^*\} :=$ extract_peak_members$(\mathbf{X}, \mathbf{Y})$
 2: index members in $X^*$ whose location has moved in the last generation
 3: find the closest other niche to each changed niche
 4: for each selected niche pair $X_i^*$ and $X_j^*$ do
 5:   if distance$(X_i^*, X_j^*) \leq tolerance$ then
 6:     merge niches
 7:   else
 8:     find the midpoints, $\mathbf{x}'$, for the paired niches
 9:     evaluate midpoint for paired niches, $y' = f(\mathbf{x}')$
10:     $evals := evals + 1$
11:     if $y' < Y_i^* \wedge y' < Y_j^*$ then
12:       add $\mathbf{x}'$ to both $\mathbf{X}_i$ and $\mathbf{X}_j$
13:     else
14:       merge niches
15:     end if
16:   end if
17: end for
18: return $\{\mathbf{X}, \mathbf{Y}, evals\}$
```

Fig. 4. The `compare_niches` subroutine.

```
Require: $\mathbf{X}, \mathbf{Y}$
 1: for each niche history set $\mathbf{X}_i$ do
 2:   $\mathbf{X}_i :=$ truncate_least_fit$(\mathbf{X}_i, \mathbf{Y}_i, 50)$
 3:   $k = min(|\mathbf{X}_i|, 50)$
 4:   if $k = 1$ then
 5:     fit $\mathcal{S}_i$ using (up to) 10 closest niche peaks in parameter space
 6:   else
 7:     fit $\mathcal{S}_i$ using $\mathbf{X}_i$ members
 8:   end if
 9: end for
10: return $\mathcal{S}$
```

Fig. 5. The `fit_local_surrogates` subroutine.

prove useful to fitting their local surrogates later). However if it is better than one or both of its parents, then the niches are merged (and the mid point added to the history of the resultant merged niche, and becomes its $\mathbf{x}^*$ estimate if appropriate).

In the `fit_local_surrogates` subroutine (Figure 5) the niche history stored for each niche is used to fit a surrogate in the region of the peak estimate. A maximum of 50 locations are stored in each peak's niche history. When the number exceeds 50, the least fit are truncated (line 2) – note, this distance is calculated from $f(\mathbf{x}^*)$ rather than $\mathbf{x}^*$. If there is only one element in the current niche, then the 10 closest other peaks in *search* space (or fewer if there are fewer than 10 niches in total), are used to fit the surrogate model (lines 4-5).

Following the fitting of the surrogate models, the surrogates are then used to evaluate proposed solutions drawn in the immediate vicinity of the niche peak, as described in Figure 6. In the special case where adjacent niches have been used to fit the surrogate, due to the niche not currently having a history of its own, then a hypersphere in $\mathcal{X}$ is placed around the niche peak. The hypersphere width is determined by the distance to the closest next niche peak (in design space), otherwise the diameter of the hypersphere placed around the peak is chosen

```
Require: $\mathcal{S}, \mathbf{X}, \mathbf{Y}, evals$
 1: for each fitted surrogate $\mathbf{X}_i$ do
 2:   $k = min(|\mathbf{X}_i|, 10)$
 3:   if $k = 1$ then
 4:     get parameter space distance $d$ to closest other niche peak
 5:   else
 6:     $u := \mathcal{U}(0, 1)$
 7:     if $u < \frac{1}{3}$ then
 8:       get parameter space distance $d$ to the closest niche history member (by fitness)
 9:     else if $u < \frac{2}{3}$ then
10:       get parameter space distance $d$ to the $k$th closest niche history member (by fitness)
11:     else
12:       get parameter space distance $d$ to closest other niche peak
13:     end if
14:   end if
15:   $u := \mathcal{U}(0, 1)$
16:   if $u < \frac{1}{2}$ then
17:     generate 100 samples in a truncated Gaussian hypersphere, centred on the $i$th peak, with the hypersphere radius set to $d/2$
18:     induce fitness estimate of samples through $\mathcal{S}_i$
19:     select sample $\mathbf{x}$ with best predicted fitness
20:   else
21:     generate a sample $\mathbf{x}$ in a truncated Gaussian hypersphere, centred on the $i$th peak, with the hypersphere radius set to $d/2$
22:   end if
23:   evaluate $\mathbf{x}$ on actual problem, to obtain $y$
24:   $evals := evals + 1$
25:   $\{\mathbf{X}, \mathbf{Y}\} :=$ update_niche$(\mathbf{X}, \mathbf{Y}, i, \mathbf{x}, y)$
26: end for
27: return $\{\mathbf{X}, \mathbf{Y}, evals\}$
```

Fig. 6. The `propose_via_surrogates` subroutine.

either as the distance to the next fittest point in the niche history (line 8), the distance to the 10th closest point in niche history (line 10) or the distance to the closest next niche peak (line 12). By varying the value chosen for the diameter $d$ in these fashion we seek to generate putative solutions which are compactly distributed about the current peak estimate for fine-tuning, and also more widely spread to make larger movements, but always within the region defined by the current niche and constrained by the next closest one. Within the hypersphere itself an isotropic Gaussian distribution is used to generate the samples (the hypersphere radius effectively defining the scaling and truncation point of the Gaussian – with three standard deviations from the hypersphere centre to its edge). 50% of the time 100 samples are drawn from this hypersphere and evaluated by the local surrogate, with the predicted fittest of these selected to be evaluated on the actual problem. The other 50% of the time we take a single random sample from the truncated Gaussian and do not fit a surrogate. By choosing the predicted fittest we are exploiting the estimated peak from the surrogate, whereas the random selection promotes *search* in the neighbourhood of the peak estimate.

```
Require: $\mathbf{X}, \mathbf{Y}, i, \mathbf{x}, y$
1: $\{\mathbf{x}^*, y^*\} :=$ get_peak($\mathbf{X}_i, \mathbf{Y}_i$)
2: if $y > y^*$ then
3:     replace peak with $\mathbf{x}$
4: end if
5: update niche history with $\mathbf{x}$ and $y$
6: return $\{\mathbf{X}, \mathbf{Y}\}$
```

Fig. 7. The `update_niche` subroutine.

```
Require: $\mathbf{X}, \mathbf{Y}, evals$
1: $\{X^*, Y^*\} :=$ extract_peak_members($\mathbf{X}, \mathbf{Y}$)
2: randomly permute the indices of the peak sets, and hold
   in $I$
3: while $|I| > 1$ do
4:     remove the last two values held in $I$, $i$ and $j$
5:     crossover $X_i^*$ and $X_j^*$ to create $\mathbf{x}'$ and $\mathbf{x}''$
6:     evaluate the offspring, $\mathbf{x}'$ and $\mathbf{x}''$
7:     create a new niche for each of the offspring
8:     $evals := evals + 2$
9: end while
10: return $\{\mathbf{X}, \mathbf{Y}, evals\}$
```

Fig. 8. The `crossover_niches` subroutine.

Before returning, the `propose_via_surrogates` subroutine passes each of the evaluated proposals to the `update_niche` subroutine, which ascertains whether the new solution has improved the peak estimate, and is described in Figure 7. If the new location is worse than the current peak, then the proposal is still added to the history of the current niche.

The main exploration driver occurs via niche crossover, as described in Figure 8 (the SBX parameter was set to 20 in our empirical work), with additional speculative search via a random element in each generation (line 14 of Figure 3).

The surrogates provide the main exploitation driver in the algorithm and crossover the principal exploration mechanism (although there is localised mutation around the peaks as described in Figure 6). Note, the algorithm does not concern itself with *how* fit the niches it maintains are in relation to each other – it is only concerned that the niches are *locally* fit with respect to the the immediate vicinity around a niche peak. As such this algorithm will successfully find many peaks of varying heights.

## V. Empirical results

The algorithm's performance is compared against the published results of a number of particle swarm optimisation (PSO) and differential evolution (DE) algorithms previously applied to multi-modal problems. Namely Constricted PSO (CPSO) [25], Gaussian PSO (GPSO) [26], Cauchy and Gaussian PSO (CGPSO) [2], DE/rand/1/bin [27] (labelled DE in the tables) and the crowding DE/nrand/1/bin [28] (labelled CDE in the tables). These results are taken from [2] and [29] – and the form of assessment follows the practice of these articles.

Eight test functions of varying dimensionality and number of optima are used, as defined in Table I, and illustrated in Figure 9, and earlier in Figure 1. The one-dimensional equal maxima ($f_1$) and uneven maxima ($f_2$) problems have the same number of global peaks (that is, niches with the same, best possible, maxima), however those in $f_2$ are not evenly spaced, and there is a niche with a lower peak located at the boundary on the left-hand side. The Himmelblau function ($f_3$) lives in two dimensions, and has four global maxima (two of these at the either end of a ridge). The six-hump camel back function ($f_4$), contrary to its name has two global peaks and two local peaks (due to the sampling region used, as in [2]). The inverted Shubert function ($f_5$) is scalable in the number of dimensions $p$ – with the number of global peaks being $p3^p$, and considerably more local peaks. The inverted Vincent function ($f_6$) has $6^p$ global peaks (but does not suffer from suboptimal local peaks). The composite functions $f_7$ and $f_8$ are non-symmetric problems with numerous local optima and are composed of the properties of a number of different functions from the literature. Due to space constraints formal definitions are not provided here – however the technical report detailing them can be found online [29].

TABLE I. Test functions.

| Source | Definition |
|---|---|
| [8] | $f_1(x) = \sin^6(5\pi x)$ |
| [8] | $f_2(x) = \sin^6\left(5\pi(x^{3/4} - 0.05)\right)$ |
| [8] | $f_3(\mathbf{x}) = 200 - (x_1^2 + x_2 - 11)^{-}(x_1 + x_2^2 - 7)^2$ |
| [9] | $f_4(\mathbf{x}) = -4\left((4 - 2.1x_1^2 + \frac{x_1^4}{3})x_1^2 + x_1x_2 + (-4 + 4x_2^2)x_2^2\right)$ |
| [6] | $f_5(\mathbf{x}) = -\prod_{i=1}^{p}\sum_{j=1}^{5} j\cos\left((j+1)x_i + j\right)$ |
| [10] | $f_6(\mathbf{x}) = \frac{1}{p}\sum_{i=1}^{p}\sin\left(10\log(x_i)\right)$ |
| [29] | $f_7(\mathbf{x}) =$ two-dimensional problem composed of six basic functions |
| [29] | $f_8(\mathbf{x}) =$ two-dimensional problem composed of eight basic functions |

TABLE II. Test problem parameters (as used in [2] and [29]).

| function | range | $\epsilon$ | $r$ | Global peak height | # global peaks |
|---|---|---|---|---|---|
| $f_1$ | $0 \le x \le 1$ | 0.0001 | 0.01 | 1.0 | 5 |
| $f_2$ | $0 \le x \le 1$ | 0.0001 | 0.01 | 1.0 | 5 |
| $f_3$ | $-6 \le x_i \le 6$ | 0.0001 | 0.01 | 200.0 | 4 |
| $f_4$ | $-1.9 \le x_1 \le 1.9$ $-1.1 \le x_2 \le 1.1$ | 0.0001 | 0.01 | 1.03163 | 2 |
| $f_5(2D)$ | $-10 \le x_i \le 10$ | 0.1 | 0.5 | 186.731 | 18 |
| $f_5(3D)$ | $-10 \le x_i \le 10$ | 0.1 | 0.5 | 2709.0935 | 81 |
| $f_5(4D)$ | $-10 \le x_i \le 10$ | 0.1 | 0.5 | 39303.55 | 324 |
| $f_6(2D)$ | $0.25 \le x_i \le 10$ | 0.01 | 0.1 | 1.0 | 36 |
| $f_6(3D)$ | $0.25 \le x_i \le 10$ | 0.01 | 0.1 | 1.0 | 216 |
| $f_6(4D)$ | $0.25 \le x_i \le 10$ | 0.01 | 0.1 | 1.0 | 1296 |
| $f_7$ | $-5 \le x_i \le 5$ | 0.01 | 0.01 | 0.0 | 6 |
| $f_8$ | $-5 \le x_i \le 5$ | 0.01 | 0.01 | 0.0 | 8 |

Table II details the variable ranges of the different test functions, and also the parameters used to assess whether an algorithm has found a peak. The value $\epsilon$ determines how close (in fitness) a solution must be to a global peak maximum to be determined to have found the peak (as long an another solution within $r$ radius in $\mathcal{X}$ has not already discovered the peak). As in [2], test problems $f_1$–$f_4$ have a maximum of 100,000 function evaluations allowed, the 2D and 3D variants of $f_5$ and $f_6$ have a maximum of 200,000 function evaluations, and the 4D variants have a maximum of 400,000 function evaluations.[2] The two composite test functions $f_7$ and $f_8$ each have a maximum of 200,000 function evaluations per run.

[2]This differs slightly from [29], where the 3D variants have a maximum of 400,000 function evaluations, and 4D variants are not investigated.

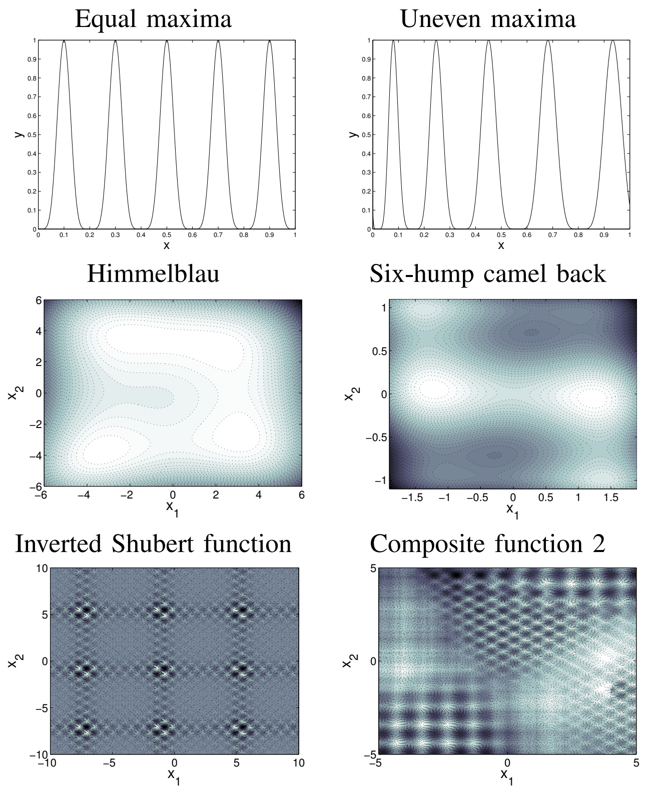


Fig. 9. Contour maps of test function landscapes. Darker greys indicate low fitness and lighter greys indicate high fitness.

TABLE III. SUCCESS RATES, INCLUDING COMPARISON RESULTS FROM [2] AND [29].

| func. | CPSO | GPSO | CGPSO | DE | CDE | LSEA |
|---|---|---|---|---|---|---|
| $f_1$ | 0.94 | 0.98 | 0.96 | **1.00** | **1.00** | **1.00** |
| $f_2$ | 0.92 | 0.94 | 0.98 | - | - | **1.00** |
| $f_3$ | 0.54 | 0.28 | 0.44 | **1.00** | 0.98 | **1.00** |
| $f_4$ | **1.00** | **1.00** | **1.00** | **1.00** | **1.00** | **1.00** |
| $f_5^{2D}$ | 0.98 | 0.76 | 0.98 | 0.00 | **1.00** | **1.00** |

TABLE IV. AVERAGE NUMBER OF FUNCTION EVALUATIONS FOR CONVERGENCE WITH ACCURACY LEVEL $\epsilon = 0.0001$ (DE VALUES FORM FROM [29]). ON OTHER TEST FUNCTIONS THE DE ALGORITHMS ALL RAN TO THE MAXIMUM NUMBER OF FUNCTION EVALUATIONS FOR THIS $\epsilon$ LEVEL.

| function | DE/nrand/1/bin | Crowding DE/rand/1/bin | LSEA |
|---|---|---|---|
| $f_1$ | 1,552 | 3,386 | 278 |
| $f_2$ | - | - | 285 |
| $f_3$ | 13,610 | 41,666 | 489 |
| $f_4$ | 3806 | 12,980 | 393 |

TABLE V. AVERAGE PEAK RATIOS, INCLUDING COMPARISON RESULTS FROM [2] AND [29]. RESULTS MARKED WITH * WERE RUN FOR TWICE AS MANY FUNCTION EVALUATIONS COMPARED TO THE OTHER ALGORITHMS.

| func. | CPSO | GPSO | CGPSO | DE | CDE | LSEA |
|---|---|---|---|---|---|---|
| $f_5^{3D}$ | 0.62 | 0.45 | 0.31 | 0.10* | 0.27* | **0.99** |
| $f_5^{4D}$ | 0.24 | 0.11 | 0.04 | - | - | **0.53** |
| $f_6^{2D}$ | 0.95 | 0.80 | **1.00** | 0.35 | 0.72 | **1.00** |
| $f_6^{3D}$ | 0.25 | 0.25 | 0.35 | ***1.00**** | ***1.00**** | **0.99** |
| $f_6^{4D}$ | 0.10 | 0.10 | 0.12 | - | - | **0.61** |
| $f_7$ | - | - | - | 0.673 | 0.69 | **0.71** |
| $f_8$ | - | - | - | **0.84** | 0.06 | 0.75 |

As in the comparison work, we run our algorithm 50 times on each problem. Two main criteria are used. On the ‘easier’ problems the *success rate* (SR) is recorded – which measures the proportion of successful runs (those which find all global optima given the prescribed $\epsilon$ and $r$). A value of 1.0 therefore indicates that all 50 runs found all global peaks, whereas a value of 0.5 would indicate that half the runs (25) found all global peaks. For the other problems, the *peak ratio* (PR) measure is employed. This gives the average proportion of global peaks found across runs, i.e. for $q$ runs:

$$PR = \frac{\sum_{i=1}^{q} d_i}{tq} \tag{1}$$

where $d_i$ denotes the number of global optima discovered by the $i$th run, and $t$ is the total number of global peaks. An SR of 1.0 also means a PR of 1.0, however the PR level cannot be deduced from an SR lower than 1.0 (as an SR of 0.0 may mean every run obtained the majority, but not all, of the peaks, or the runs may have found no peaks whatsoever). A ‘-’ in a table entry in Tables III-V indicates where an algorithm was not run on that particular problem in the comparison studies.

For test problems $f_1, f_2, f_3, f_4$ and $f_5^{2D}$ the success rates of the localised surrogates evolutionary algorithm (LSEA) compared well to the other algorithms (results shown in Table III). LSEA finds the peaks consistently, with the DE approaches finding all peaks on three problems for all 50 runs – however as shown in Table IV, the LSEA approach obtains equivalent or better results in many fewer function evaluations on these problems.

For the more difficult problems the average peak ratio values in can be found in Table V. Direct comparison with the some of the DE results is a little more difficult here, as [29] does not report results for the $4D$ variants, and they were run for twice as many function evaluations than the other algorithms for the 3D variants. For both the $f_5$ and $f_6$ problems LSEA is seen to perform well in comparison to the other algorithms. Overall LSEA is seen to perform equivalently or better than the other algorithms on 11 out of the 12 test problem formulations. Even given this, its performance does seem to degrade when a problem has an excessive number of local peaks (global or local). We will now examine the behaviour of the population maintained by LSEA to see why this occurs, and how LSEA may be developed further.

## *A. Examination of the niche maintenance dynamics within the proposed algorithm*

The effect of the dynamic niche maintenance used in LSEA can be seen when looking at the number of niches maintained by the algorithm on the different problems across the 50 runs. Figure 10 shows the number of estimated niches maintained for each run of LSEA for each of the problems (until the point at which they find all global optima, or complete the maximum permitted function evaluations). The number of ‘true’ global optima for each problem is plotted as a horizontal dotted line. The spike of new niches generated via crossover every five generations is clear to see in the panels with the population doubling, and then retracting as a number are merged in the subsequent generation).

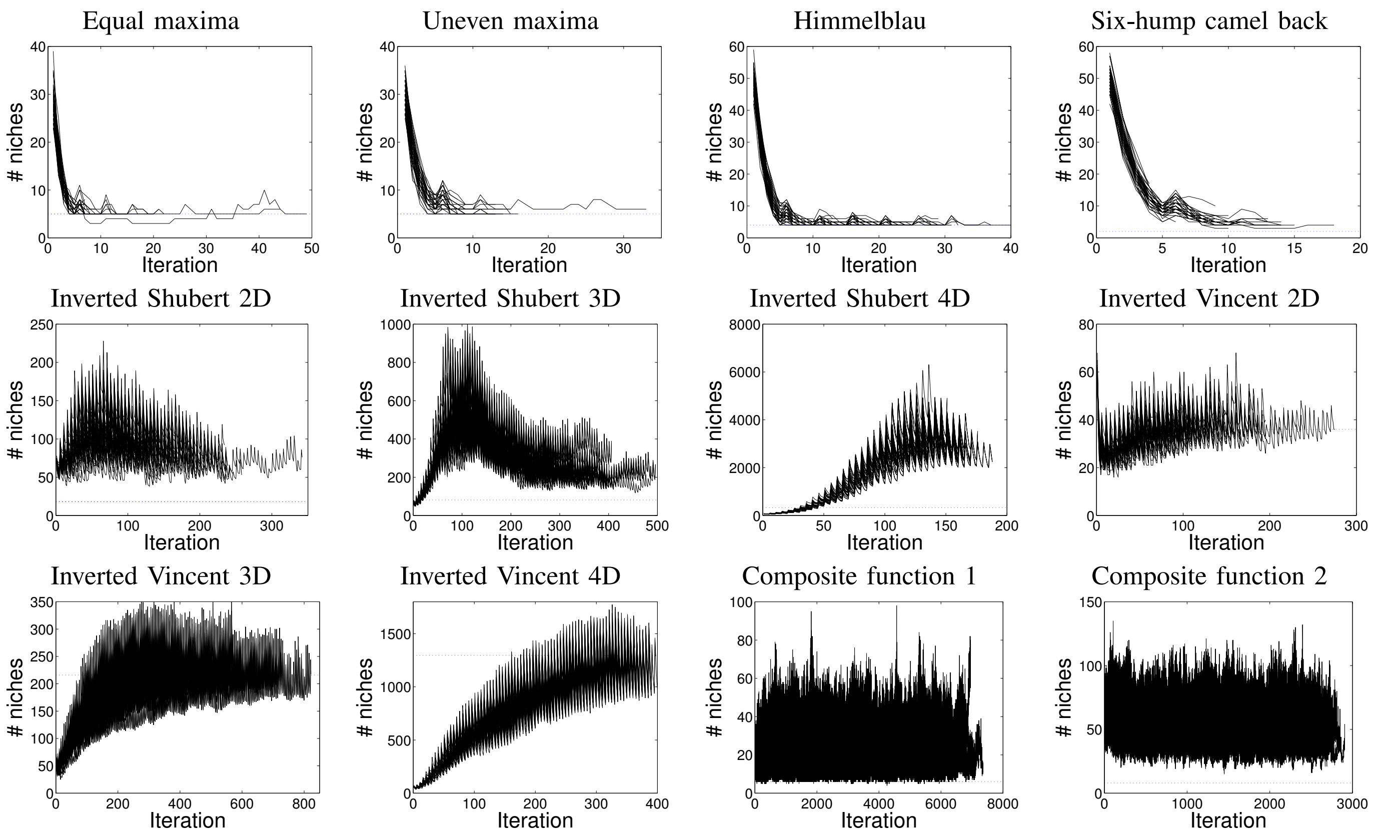


Fig. 10. Maintained niche size per generation for each run (recorded after line 6 of Figure 3). The dotted horizontal line shows the total number of global optima for the problem. Population shown until all niches are found (or until all permitted function evaluations are exhausted).

For simpler test functions the number of niches maintained rapidly converges to the actual number of niches. For example, in the equal maxima function panel (top left of Figure 10), nearly all runs have converged within 20 iterations, with one of the runs taking a little longer as it 'undershoots' with niche merging and needs to rediscover the region of the final niche.

For more complex test functions with local optima, the number of niches maintained can be seen to rapidly expand to a level considerably above the number of global optima, before dropping back to a level where it is searching the global niche regions, but also a good number of the local niches. In the case of local optima living on a larger scale landscape feature (as seen in composite function 1), the local optima do not seem to be regularly maintained – this is likely due to the mid-point merging in the `compare_niches` subroutine 'smearing' these local optima out in favour of optimising the larger landscape feature they reside on. However local optima which do not lie on a larger scale landscape features tend to be maintained. This is contrast is illustrated in Figure 11, which plots the niche peak estimate locations found after 200,000 function evaluations of the inverted Shubert function and the composite function 1.

Although the algorithm is seen to perform well against existing state-of-the-art optimisers, by examining *how* the niche population is evolving there does appear to be scope for further improvement. Avenues of further investigation are:

1) Using more sophisticated analysis when merging niches or keeping them separate — currently a basic check is undertaken based on the fitness of a midpoint between two locations, however a surrogate could also be employed here (although it may require storing the less fit solutions at the edge of a niche, as well as those near the current local maxima).
2) Integrate methods to smooth away local optima to mitigate their negative impact on the search (as in for instance [15], [16]).
3) Bias the search toward peaks with higher fitness (currently all niches are treated equally when found).

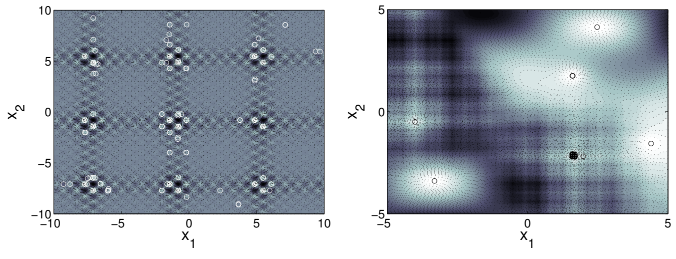


Fig. 11. Niche peak locations returned by LSEA after 200,000 function evaluations (marked with circles) on the 2D inverted Shubert function (left panel) and the first composite function (right panel). Many more *local* modes are seen to be maintained for the former due to the landscape features.

In addition to the above, it may be useful to explore the use of other surrogate models, and examine the effect of the *tolerance* value used for merging niches. Also there is a potential to develop a heuristic for convergence checking on a particular niche (i.e., exploring when to stop evolving a niche if it is thought the best value in that local region has now been attained).

Finally, the use of many local surrogates rather than a single global surrogate has large efficiency gains in model fitting, as well as fidelity, however there is still a cost overhead in evaluating proposals with a surrogate. So, for cheap to compute $f(\cdot)$ there may be a trade-off in the number of surrogate evaluations to use per iteration when compared to the number of evaluations of $f(\cdot)$ taken.

## VI. Discussion

A new approach to multi-modal optimisation based on using an EA with a dynamically varying number of localised surrogates has been proposed, and evaluated on a range of problems. It may be viewed as a sophisticated distributed hill-climber, albeit one employing a degree of communication between immediately adjacent hills, which uses local surrogates to guide the hill traversal. The EA component is principally concerned with searching for *new* hills to climb, although it will also exploit any symmetry in the landscape. It is however seen to perform well on problems which are non-symmetric too.

Analysis of its behaviour indicates there are areas that can still be improved, with efficiency gains still to be had by improving its niche maintenance subroutines, and how it handles problems with very many local optima. Nevertheless, even in its current form LSEA is found to be competitive with the state-of-the-art.

## Acknowledgments

The author would like to thank Xiaodong Li for inspiring him to consider the multi-modal optimisation area anew, and kindly discussing the state of research in the field during the IEEE WCCI 2010 conference. Thanks also to the anonymous referees for their valuable comments and helpful suggestions.

## References

[1] B. Sareni and L. Krähenbühl, "Fitness sharing and niching methods revisited," *IEEE Transactions on Evolutionary Computation*, vol. 2, no. 3, pp. 97–106, 1998.

[2] X. Li and K. Deb, "Comparing lbest PSO niching algorithms using different position update rules," in *IEEE Congress on Evolutionary Computation*, 2010, pp. 1564–1571.

[3] X. Yin and N. Germany, "A fast genetic algorithm with sharing scheme using cluster analysis methods in multi-modal function optimization," in *International Conference on Artificial Neural Networks and Genetic Algorithms*, 1993, pp. 450–457.

[4] D. Beasley, D. R. Bull, and R. Martin, "A sequential niche technique for multimodal function optimization," *Evolutionary Computation*, vol. 1, no. 2, pp. 101–125, 1993.

[5] G. Harik, "Finding multimodal solutions using restricted tournament selection," in *Proceedings of the Sixth International Conference on Genetic Algorithms*, 1995, pp. 24–31.

[6] J.-P. Li, M. Balazs, G. Parks, and P. Clarkson, "A species conserving genetic algorithm for multimodal function optimisation," *Evolutionary Computation*, vol. 10, no. 2, pp. 207–234, 2002.

[7] K. Parsopoulos and M. Vrahatis, "On the computation of all global minimizers through particle swarm optimization," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 211–224, 2004.

[8] K. Deb, "Genetic algorithms in multimodal function optimization," Masters thesis and TCGA report no. 89002, Tuscaloosa: University of Alabama, The Clearinghouse for Genetic Algorithms, Tech. Rep., 1989.

[9] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, 1996.

[10] O. Shir and T. Bäck, "Niche radius adaptation in the CMS-ES niching algorithm," in *Parallel Problem Solving from Nature - PPSN IX*, ser. LNCS, vol. 4193. Springer, 2006, pp. 142–151.

[11] K.-H. Liang, X. Yao, and C. Newton, "Evolutionary Search of Approximated N-Dimensional Landscapes," *International Journal of Knowledge-Based Intelligent Engineering Systems*, vol. 4, no. 3, pp. 172–183, 2000.

[12] J. Holland, *Adaptation in Natural and Artificial Systems*. University of Michigan Press, 1975.

[13] D. E. Goldberg and J. Richardson, "Genetic algorithms with sharing for multimodal function optimisation," in *Proceedings of the Second International Conference on Genetic Algorithms and their Application*. L. Erlbaum Associates Inc., 1987, pp. 41–49.

[14] Y. Jin, "Surrogate-assisted evolutionary computation: Recent advances and future challenges," *Swarm and Evolutionary Computation*, vol. 1, no. 2, pp. 61–70, 2011.

[15] D. Yang and S. J. Flockton, "Evolutionary Algorithms with a Coarse-to-Fine Function Smoothing," in *IEEE International Conference on Evolutionary Computation*. IEEE Press, 1995, pp. 657–662.

[16] D. Lim, Y. Jin, Y.-S. Ong, and B. Sendhoff, "Generalizing Surrogate-Assisted Evolutionary Computation," *IEEE Transactions on Evolutionary Computation*, vol. 14, no. 3, pp. 329–355, 2010.

[17] H.-M. Voight and J. M. Lange, "Local Evolutionary Search by Random Memorizing," in *IEEE International Conference on Evolutionary Computation*. IEEE Press, 1998, pp. 547–552.

[18] X. Lu, K. Tang, and X. Yao, "Classification-assisted Differential Evolution for computationally expensive problems," in *IEEE Congress on Evolutionary Computation*, 2011, pp. 1986–1993.

[19] Y. Tenne and S. W. Armfield, "A framework for memetic optimization using variable global and local surrogate models," *Soft Computing*, vol. 13, pp. 781–793, 2009.

[20] Y. Jin, M. Olhofer, and B. Sendhoff, "A framework for evolutionary optimization with approximate fitness functions," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 5, pp. 481–494, 2002.

[21] J. Sacks, W. J. Welch, T. J. Mitchell, and H. P. Wynn, "Design and analysis of computer experiments," *Statistical Science*, vol. 4, no. 4, pp. 409–435, 1989.

[22] S. N. Lophaven, H. B. Nielsen, and J. Søndergaard, "DACE A Matlab Kriging Toolbox," Technical Report IMM-TR-2002-12, Informatics and Mathematical Modelling, Technical University of Denmark, Tech. Rep., 2002.

[23] C. Rasmussen and C. Williams, *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[24] K. Deb and R. B. Agrawal, "Simulated Binary Crossover for Continuos Search Space," Department of Mechanical Engineering, Indian Institute of Technology, Kanpur, IITK/ME/SMD-94027, Tech. Rep., 1994.

[25] M. Clerc and J. Kennedy, "The particle swarm - explosion, stability, and convergence in a multidimensional complex space," *IEEE Transactions on Evolutionary Computation*, vol. 6, pp. 58–73, 2002.

[26] B. Secrest and G. Lamont, "Visualizing particle swarm optimization - gaussian particle swarm optimization," in *Proceedings of the 2003 IEEE Swarm Intelligence Symposium*, 2003, pp. 198–204.

[27] R. Thomsen, "Multimodal optimization using crowding-based differential evolution," in *IEEE Congress on Evolutionary Computation*, 2004, pp. 1382–1389.

[28] M. Epitropakis, V. Plagianakos, and M. Vrahatis, "Finding multiple global optima exploiting differential evolution's niching capability," in *IEEE Symposium on Differential Evolution (IEEE Symposium Series on Computational Intelligence)*, 2011, pp. 80–87.

[29] X. Li, A. Engelbrecht, and M. Epitropakis, "Benchmark Functions for CEC'2013 Special Session and Competition on Niching Methods for Multimodal Function Optimization," Evolutionary Computation and Machine Learning Group, RMIT University, Tech. Rep., 2013.