Running Up Those Hills: Multi-Modal Search with the Niching Migratory Multi-Swarm Optimiser

Jonathan E. Fieldsend
College of Engineering, Mathematics and Physical Sciences
University of Exeter
Exeter, UK, EX4 4QF
Email: J.E.Fieldsend@exeter.ac.uk

***Abstract*—We present a new multi-modal evolutionary optimiser, the niching migratory multi-swarm optimiser (NMMSO), which dynamically manages many particle swarms. These sub-swarms are concerned with optimising separate local modes, and employ measures to allow swarm elements to migrate away from their parent swarm if they are identified as being in the vicinity of a separate peak, and to merge swarms together if they are identified as being concerned with the same peak. We employ coarse peak identification to facilitate the mode identification required. Swarm members are not constrained to particular sub-regions of the parameter space, however members are initialised in the vicinity of a swarm's local mode estimate. NMMSO is shown to cope with a range of problem types, and to produce results competitive with the state-of-the-art on the CEC 2013 multi-modal optimisation competition test problems, providing new benchmark results in the field.**

I. Introduction

An effective multi-modal optimiser embodies a number of traits: (1) it is dynamic in the number of modes it maintains and returns, enabling it to be applied to problems with few or many modes; (2) it is self-adaptive, with few meta-parameters, enabling a wide range of problem types to be tackled without prior tuning; (3) it incorporates exploitative local search, enabling it to rapidly hone the peak estimates it is maintaining. Although the algorithm genus may vary (genetic algorithm, evolutionary strategy, particle swarm, differential evolution, etc.), it is these three properties which all the best-performing algorithms in the CEC 2013 competition contain [1]. Here we present a new optimiser which embeds the three properties listed above, and utilises multiple particle swarm optimisers to rapidly climb peaks in the search landscape. The swarms do not operate in isolation: elements can migrate away from their parent swarm if they are judged to have discovered a separate peak. Swarms may also be *merged* if they are identified as converging on the same peak. Additionally, swarm members are not constricted to movements within a local region of design space.

The paper proceeds are follows. In Section II we describe the general multi-modal optimisation problem, and highlight the difficulties that confront optimisers in this arena. In Section III we briefly discuss some of the popular evolutionary computation approaches to multi-modal optimisation. In Section IV we introduce the niching migratory multi-swarm optimiser (NMMSO), describing its properties and relationship to other approaches. This is followed by empirical results on the CEC 2013 competition test problems. The paper ends with a discussion in Section VI.

II. Multi-modal optimisation

The general aim in multi-modal optimisation is similar to standard uni-objective optimisation, that is, given a legal search domain $\mathcal{X}$, without loss of generality, we seek to maximise $f(\mathbf{x})$, $\mathbf{x} \in \mathcal{X}$, given any equality and inequality constraints. In the case of a multi-modal problem however, we seek not simply to discover a single design $\mathbf{x}$ which maximises $f(\mathbf{x})$ given the constraints, but *all* $\mathbf{x}^* \in \mathcal{X}$ which obtain the maximum 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 equal or lower than $f(\mathbf{x}^*)$. *Local* optima (local modes/peaks) in contrast 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 obtainable. Local regions around a peak are often called *niches*.

There are many reasons that the problem owner may wish multiple mode solutions to be discovered rather than a single 'best' solution. By discovering a range of different designs which are operationally equivalent insight into the problem domain may be extracted. Also, it may transpire that some designs are not machinable – i.e., $\mathcal{X}$ is misspecified, and therefore a range of solutions mitigates against this. Finally $f(\cdot)$ may be in error in certain regions, therefore a wide range of good solutions can be helpful if the 'best' design does not perform as emulated (it is useful to have local optima, not just global optima stored – as there is no guarantee that all the global optima under $f(\cdot)$ are not in error).

III. Evolutionary multi-modal optimisers

One of the earliest approaches to evolutionary multi-modal optimisation is derived from the fitness sharing concept, first introduced in [2]. This was later refined as a means to partition a genetic algorithm search population into different subpopulations based on their fitness [3]. An overview of these general niching ideas is presented in [4].

As highlighted in e.g. [5], many early multi-modal optimisers tended to be highly parametrised, relying on well-chosen values to perform well (for instance specifying *a priori* what

the niche width should be set as, or how many modes to search for). However, a recent design trend of the most effective multi-modal optimisers is to make them to a large degree 'self-tuning'.

The current state-of-the-art (based upon the results of the CEC 2013 competition in the field) rely on a range of different technologies and heuristics to maintain, search for and exploit mode estimates. The optimiser that was the best performing overall, proposed in [6], utilises the covariance matrix adaptation evolution strategy (CMA-ES) of [7]. Rather than selecting the restart location at random, [6] used nearest-better clustering to partition a search population into sub groups concerned with different modes. This is facilitated by fitting a spanning tree on the population, linking all population members to their nearest neighbour (in design space) which is better performing under $f(\cdot)$, and disconnecting the longest edges (thus assuming that the best search points on different peaks are likely to be further away from each other than neighbours on the same peak). This leads to another property of this approach – it is *dynamic* in the number of modes it maintains and returns (although limited to a maximum number, set *a priori*). The second best performing multi-modal optimiser [8] also has dynamic mode maintenance, storing an external dynamic archive of estimated mode locations which supports a reinitialisation mechanism, along with an adaptive control parameter technique for the differential evolution algorithm driving the search. The third best was the standard CMA-ES algorithm. Finally, the fourth ranked algorithm proposed by [9] uses an external memory to store the current global optima, along with an adaptive niche radius to mitigate the effect of setting this parameter *a priori* to a value which may not be appropriate to the problem at hand. A mesh of solutions is exploited, with a combination method that generates solutions in the direct of nearest (estimated) global optima.

The published ranking of these algorithms is derived from their average performance on the twenty problem formulations used in the CEC 2013 benchmark suite, averaged across five different accuracy levels for fixed numbers of function evaluations. Given different test problems, and/or a different number of permitted function evaluations and accuracy levels, there may of course be a different ranking obtained. The top ranked algorithms do however possess a number of similar characteristics which would seem to describe an effective multi-modal optimiser, namely: self-adaption of search parameters, dynamic mode maintenance, and exploitative local search. Here we leverage and develop these ideas, and build on aspects of our recent work in the area, which use coarse mode region identification in conjunction with local surrogates and hill-climbers [10], [11].

IV. Using multiple swarms

Particle swarm optimisation PSO has gained widespread popularity since its introduction in [12] for the optimisation of continuous non-linear functions, due to its rapid convergence properties on a wide range of problems, and its relative simplicity (and therefore ease of implementation). A fixed population of solutions is used, where each solution (or particle) is represented by a point in D-dimensional design space. The ith particle is commonly represented as $\mathbf{x}_i = (x_{i,1}, x_{i,2}, \dots x_{i,D})$, and its performance is evaluated on a given problem and stored. Each particle maintains knowledge of its best previous evaluated position, represented as $\mathbf{p}_i$ (commonly referred to as 'pbest'), and also has knowledge of the single best solution found so far in some defined neighbourhood, $\mathbf{g}_i$ (commonly referred to as 'gbest'). Often this is with respect to a global neighbourhood (all particles are considered), however other neighbourhood definitions can also be used. The rate of position change of a particle then depends upon its previous personal best position and the neighbourhood best, and its previous velocity. For particle i this velocity is $\mathbf{v}_i = (v_{i,1}, \dots, v_{i,D})$, typically initialised at random in $\mathcal{X}$. The general algorithm for the adjustment of these velocities is:

$$v_{i,j} := \omega v_{i,j} + c_1 r_1 (b_{i,j} - x_{i,j}) + c_2 r_2 (g_{i,j} - x_{i,j}), \quad (1)$$

and the position is updated as:

$$x_{i,j} := x_{i,j} + \chi v_{i,j}, \quad j = 1, \dots, D, \quad (2)$$

where ω, c_1, c_2, $\chi \geq 0$. ω is the inertia of a particle, c_1 and c_2 are constraints on the velocity toward local best and neighbourhood best - referred to as the *cognitive* and *social* learning factors respectively, χ is a constraint on the overall shift in position, and $r_1, r_2 \sim \mathcal{U}(0, 1)$. In [12], the final model presented has w and χ set at 1.0 and c_1 and c_2 set at 2.0.

As discussed in [13], in this classical form of PSO each particle $\mathbf{x}_i$ is flown toward $\mathbf{p}_i$, $\mathbf{g}_i$ and $\mathbf{v}_i$. This, in effect, means that a hypercuboid is generated in design space, the bounds of which are the sum of the distances from $\mathbf{x}_i$ to the other three points (weighted by the appropriate multiplier constants from (1) and (2)). The length of the jth dimension of the containing hypercuboid of $\mathbf{x}_i$ is:

$$l_j = \chi(w v_{i,j} + c_1 (b_{i,j} - x_{i,j}) + c_2 (g_{i,j} - x_{i,j})). \quad (3)$$

A particle $\mathbf{x}_i$ can therefore effectively move to any point within this hypercuboid (determined by the draws of $\mathbf{r}_1$ and $\mathbf{r}_2$), but not *outside* of it. Note – depending on the values of χ, c_1 and c_2, it is possible for one or more of $\mathbf{v}_i$, $\mathbf{p}_i$ and $\mathbf{g}_i$ to lie outside this bounded region, and the higher the inertia, the more exploratory the search. As regions of the hypercuboid may lie outside of $\mathcal{X}$, a PSO implementation must have a mechanism to deal with potential movements outside the legal bounds. We use rejection sampling here – i.e. $\mathbf{r}_1$ and $\mathbf{r}_2$ are resampled until a legal $\mathbf{x}_i$ results.

Here we exploit the swarm paradigm for multi-modal optimisation. However, instead of employing a single swarm optimiser and using neighbourhood topology to maintain modes (e.g. [5]), we use *multiple* swarms – each of which is concerned with optimising a particular mode estimate that has been identified in the search landscape. The approach taken differs from other multi-swarm work for multi-modal problems (e.g. [14]), in that the sub-swarms do not take a "devour and move on approach", but instead concern themselves solely

with the improvement of their local peak estimate, from the time of that particular swarm's initialisation, until the algorithm termination (or its merging with another swarm). Unlike other sub-swarm work which use distributed swarms (e.g. [15]) the number of swarms is *dynamic*, and expends far fewer evaluations on niche detection.

The basic idea is that NMMSO manages a number of swarms which have strong local search, and which 'fine-tune' their local mode estimates each generation. Additionally, on each generation swarms which have improved their mode estimate are paired with their closest adjacent swarm to see if they should merge (thus preventing duplication of labour). New regions in which to search for modes are identified by splitting away particles from existing swarms, and via random search and crossover.

A high level description of the algorithm can be found in Figure 1. The algorithm takes in four overarching parameters: max_evals, tol, n and max_inc, alongside the standard PSO parameters which are used by all the sub-swarms. The parameter max_evals sets the total number of permissible function evaluations. tol is a small tolerance value, which specifies the Euclidean distance in design space where two peak estimates (and associated swarms) are automatically merged (we use 10^{-6} in our experiments here). n is the maximum number of particles in any swarm, and max_inc is the maximum number of swarms to increment per algorithm iteration.

The algorithm starts by generating and evaluating a single solution at random within $\mathcal{X}$, making the initial swarm (Figure 1, line 1). The algorithm then continues in an optimisation loop, until the allocated function evaluations are exhausted. Line 4 checks if any swarms are flagged. This will be the case if their gbest (mode estimate) has changed in the last algorithm iteration, or if the swarm has just resulted from the merging of two previous swarms. Those swarms that have been marked are compared to their nearest neighbour (based on the Euclidean distance between their respective gbests locations in design space). If the nearest neighbour is within tol distance, then the swarms are automatically merged, if not, then the mid-point in design space between the gbest locations is evaluated. If this location is worse than both of the swarm gbests, the pair are maintained separately – if not they are merged. The variable m tracks how many mid-points have been evaluated each time the routine is called. Sampling along a line for peak detection is not new (being introduced in [16]) – however typically multiple points are sampled each time, and the detection procedure is regularly undertaken (e.g. [16], [15]), consuming the majority of function evaluations during an optimisation. In contrast, NMMSO uses only a small fraction of its function evaluations on line sampling.

When merging swarms, if the total number of particles in both swarms is less than, or equal to, the swarm limit n, then the resultant swarm simply contains all the elements of both swarms. If however the total number of elements in both exceeds n, then the fittest n particles across both swarms are used to create the merged swarm, with the remaining particles being discarded.

Require: $max\_evals, tol, n, max\_inc, c_1, c_2, \chi, \omega$
1: $\mathcal{S} := \texttt{initialise\_swarm}(1)$
2: $evals := 1$
3: **while** $evals < max\_evals$ **do**
4: **while** $\texttt{flagged\_swarms}(\mathcal{S}) = \texttt{true}$ **do**
5: $\{\mathcal{S}, m\} := \texttt{attempt\_merge}(\mathcal{S}, n, tol)$
6: $evals := evals + m$
7: **end while**
8: $\mathcal{S} := \texttt{increment}(\mathcal{S}, n, max\_inc, c_1, c_2, \chi, \omega)$
9: $evals := evals + \texttt{min}(|\mathcal{S}|, max\_inc)$
10: $\{\mathcal{S}, k\} := \texttt{attempt\_separation}(\mathcal{S}, tol)$
11: $evals := evals + k$
12: $\mathcal{S} := \texttt{add\_new\_swarm}(\mathcal{S})$
13: $evals := evals + 1$
14: **end while**
15: $\{X^*, Y^*\} := \texttt{extract\_gbest}(\mathcal{S})$
16: **return** X^*, Y^*

Fig. 1. High-level pseudocode of NMMSO.

The evaluation of a mid-point between two mode estimate locations is a coarse way of mode region identification. There are a number of landscape conditions where it will identify two solutions as lying on different peaks when they are in fact on the same mode. This may occur if there is a ridge curving away and then back joining the two points, or when a point 'lower down' on the same peak region is closer to a point on *another* mode, than a point lying 'further up' on the same mode. This means the point lower down will be paired for comparison with the point on the other mode, resulting in both the mode estimates lying on different parts of the same peak region being maintained as gbests for distinct swarms. However, as the swarms optimise and improve their gbests (move up the peaks), mode estimates will move to positions where their relationship is correctly identified (resulting in merging).

After merging has been attempted, each swarm is incremented (line 8). If a swarm has fewer than n particles, then a single new particle is added to the swarm and evaluated. This new entrant is sampled uniformly in a hypersphere centred on the swarm's current peak estimate (its gbest), with the radius corresponding to half the distance to the nearest neighbouring swarm peak estimate (subject to the sample lying in $\mathcal{X}$). The velocity is sampled in a similar fashion (but centred on $\mathbf{0}$, without having to lie in $\mathcal{X}$). Note that although the particles are initialised within this local region, their subsequent movement is not restricted (beyond staying in $\mathcal{X}$). If a swarm has reached its full quota of n particles, then on incrementing the swarm one of its particles is selected at random, and updated according to (1) and (2). The number of swarms incremented each generation is limited to max_inc – if the number of swarms maintained exceeds this value, then 50% of the time the max_inc swarms with the best gbests are incremented, the other 50% of the time max_inc swarms are selected at random for incrementing. This limit is to prevent

TABLE I
PARAMETERS USED FOR PERFORMANCE MEASUREMENT, AND MAXIMUM NUMBER OF FUNCTION EVALUATIONS PER OPTIMISER RUN.

Function	F1	F2	F3	F4	F5
r	0.01	0.01	0.01	0.01	0.5
Peak height	200	1	1	200	1.03163
Max evals	50k	50k	50k	50k	50k
Function	F6	F7	F8	F9	F10
r	0.5	0.2	0.5	0.2	0.01
Peak height	186.731	1	2709.0935	1	-2
Max evals	200k	200k	400k	400k	200k
Function	F11	F12	F13	F14	F15
r	0.01	0.01	0.01	0.01	0.01
Peak height	0	0	0	0	0
Max evals	200k	200k	200k	400k	400k
Function	F16	F17	F18	F19	F20
r	0.01	0.01	0.01	0.01	0.01
Peak height	0	0	0	0	0
Max evals	400k	400k	400k	400k	400k

the algorithm exhausting too many function evaluations on poor performing local modes. It biases the search toward the best max_inc found so far, but still searches in the wider population of swarms, as these swarms may find substantially better solutions at a later point in time and may also provide useful particles when merging.

On line 10 a single swarm is selected at random from those which are at capacity (i.e. have n elements). This is checked to see if it should have a particle removed to seed a new swarm. First a particle, $\mathbf{x}_i$, is selected at random from the swarm. If $\mathbf{x}_i$ is more than tol distance from the swarm's $gbest$, then the mid-point between it and the swam gbest is evaluated.[1] If the performance of this mid-point is worse than $f(\mathbf{x}_i)$, then $\mathbf{x}_i$ is removed from its parent swarm, and used to seed a new swarm, taking its velocity with it. It is replaced in its previous swarm by the evaluated mid-point, whose velocity is initialised at random as if it was a new swarm entrant. If however the mid-point is not worse than $\mathbf{x}_i$, then $\mathbf{x}_i$ remains in its original swarm, and instead the swarm's gbest is compared to the mid-point evaluation: if it is worse, then the gbest is replaced. Very early in an optimiser run, as none of the maintained swarms has reached capacity, no mid-points are evaluated due to the `attempt_separation` routine (i.e. $k = 0$). As the optimisation progresses, typically $k = 1$ at each iteration, although as swarms converge on their gbest locations the randomly selected $\mathbf{x}_i$ may be within the tol distance, and so $k = 0$ may still occur later in an optimisation run.

Finally, each algorithm iteration ends with a new swarm being generated. 50% of the time this is seeded with a uniformly sampled location in $\mathcal{X}$, otherwise it is seeded by a single offspring solution, generated using uniform crossover of two randomly chosen swarm gbests (the probability of inheriting a particular design parameter from either parent being 0.5). The generation of random swarms means the algorithm is constantly looking for new peaks in addition to those currently being optimised, and the uniform crossover means any peak symmetry in the landscape is exploited.

[1]If it is less than tol distance away, even if it is split off, it will be merged back into the swarm on line 5 of the next iteration, so it is computationally efficient to check this beforehand.

V. EMPIRICAL RESULTS

The NMMSO algorithm is compared to the results of a wide range of multi-modal optimisation algorithms [17], [7], [18], [8], [19], [20], [21], [6], [9], [22], which were applied to the 20 benchmark problems of the CEC 2013 "Competition on Niching Methods for Mutlimodal Optimization" [23], [1]. We follow the algorithm assessment protocol used in the CEC 2013 competition; our results can therefore be directly compared to the combined competition results.

We also compare to the Multinational Evolutionary Algorithm (MEA) [16] and Multi-Sub-Swarm Particle Swarm Optimisation Algorithm (MSSPSO) [15] – as these both embed 'hill-valley' approaches to niche maintenance. Parameters for these comparison algorithms are fixed as in their respective original publications. In [16] the population size is varied across test problems from 750-1500, we use 1000 for all test problems here. Likewise in [15] population size varied from 1-40 and number of sub-swarms from 5-25. In keeping with the original competition, algorithm parameters are fixed for all problems – a population size of 20 and 25 sub-swarms is used here for MSSPSO.[2]

The 20 benchmark problems are of varying dimensionality and number of optima, and are derived from 12 base test problems. The Equal Maxima (F2), Himmelblau (F4), Vincent (F7, F9) and Modified Rastrigin (F10) problems *only* have global peaks. The Five-Uneven-Peak Trap (F1), Uneven Decreasing Maxima (F3), Shubert (F6, F8), and Composite Functions 1 (F11), 2 (F12), 3 (F13, F14, F16, F18) and 4 (F15, F16, F19, F20) all have local maxima as well as global maxima (with the Shubert and Composite Functions having *many* more local maxima than global maxima). Due to space constraints formal definitions are not provided here – however a technical report detailing them can be found online [23].

Additionally, NMMSO is compared to the problem level results which are published in [9] and [8] for the niching variable mesh optimisation (N-VMO) and the dynamic archiving niching differential evolution (dADE) algorithms.

Problem assessment criteria are detailed in Table I. The parameter r gives the maximum distance (in design space) a solution may be from a peak be categorised to have found it – subject to a further *accuracy* level, ϵ, which gives the maximum distance from the global maximum in objective space. For all problems five different accuracy levels are assessed, $\epsilon = \{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}\}$.

All algorithms are run 50 times on each problem. Two criteria are used for assessment. The *success rate* (SR) measures the proportion of successful runs (those which find *all* global optima given the prescribed ϵ and r). A value of 1.0 indicates that all 50 runs found all global peaks, whereas a value of 0.5 indicates that half of the runs found all global peaks. The *peak ratio* (PR) measure gives the average proportion of global

[2]The value of the penalty applied to swarm members who stray onto other peaks is not detailed in [15]. Here we set it such that straying particles cannot replace their pbest.

TABLE II
CONVERGENCE RATES OF NMMSO, N-VMO AND DADE/NRAND/1/BIN. BEST VALUES FOR EACH PROBLEM ARE UNDERLINED.

$\epsilon = 10^{-1}$											
Algorithm	Function	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10
dADE/nrand/1	Mean	5922	221	203	3107	367	**27459**	**2911**	**367282**	**396811**	3392
	St. D.	1673	38	14	845	121	6904	618	42450	22547	653
NMMSO	Mean	**578**	**167**	**46**	**1191**	**114**	68441	27349	391589	399982	**1422**
	St. D.	135	62	46	947	53	42978	14198	21682	125	444

Algorithm	Function	F11	F12	F13	F14	F15	F16	F17	F18	F19	F20
dADE/nrand/1	Mean	145456	114735	182185	**219869**	**61965**	**292773**	**200503**	**392376**	**340214**	400000
	St. D.	58240	21749	47527	154230	16890	133136	127782	33324	95904	0
NMMSO	Mean	**5836**	**49537**	**53038**	391400	400000	400000	400000	400000	400000	400000
	St. D.	2872	41905	40697	36202	0	0	0	0	0	0

$\epsilon = 10^{-4}$											
Algorithm	Function	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10
dADE/nrand/1	Mean	20202	1801	1290	12703	3567	150328	200000	393667	400000	12904
	St. D.	2788	586	565	1668	652	35209	0	17665	0	2169
N-VMO	Mean	12795	31835	380	25769	13012	200000	200000	400000	400000	181772
	St. D.	236	2847	128	265	278	0	0	0	0	24688
NMMSO	Mean	**1089**	**487**	**342**	**1910**	**617**	**88759**	**31350**	**392525**	400000	**2686**
	St. D.	179	124	131	913	149	40262	13175	21587	0	388

Algorithm	Function	F11	F12	F13	F14	F15	F16	F17	F18	F19	F20
dADE/nrand/1	Mean	200000	200000	200000	400000	400000	400000	400000	400000	400000	400000
	St. D.	0	0	0	0	0	0	0	0	0	0
N-VMO	Mean	200000	200000	200000	400000	400000	400000	400000	400000	400000	400000
	St. D.	0	0	0	0	0	0	0	0	0	0
NMMSO	Mean	**9057**	**68049**	**74120**	400000	400000	400000	400000	400000	400000	400000
	St. D.	3261	38807	43854	0	0	0	0	0	0	0

TABLE III
SUCCESS RATES OF NMMSO, N-VMO AND DADE/NRAND/1/BIN. BEST VALUES FOR EACH PROBLEM AND ACCURACY LEVEL ARE UNDERLINED.

	F1			F2			F3			F4			F5		
ϵ	NMMSO	N-VMO	dADE	NMMSO	N-VMO	dADE	NMMSO	N-VMO	dADE	NMMSO	N-VMO	dADE	NMMSO	N-VMO	dADE
10^{-1}	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
10^{-2}	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
10^{-3}	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
10^{-4}	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
10^{-5}	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
	F6			F7			F8			F9			F10		
ϵ	NMMSO	N-VMO	dADE	NMMSO	N-VMO	dADE	NMMSO	N-VMO	dADE	NMMSO	N-VMO	dADE	NMMSO	N-VMO	dADE
10^{-1}	0.960	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**0.260**	0.000	0.020	0.020	**1.000**	**1.000**	**1.000**	**1.000**	0.500
10^{-2}	0.960	**1.000**	**1.000**	**1.000**	**1.000**	0.240	**0.220**	0.000	0.000	0.000	0.000	**1.000**	**1.000**	**1.000**	0.380
10^{-3}	0.960	0.360	**1.000**	**1.000**	0.140	0.020	**0.180**	0.000	0.000	0.000	0.000	**1.000**	**1.000**	**1.000**	0.280
10^{-4}	**0.940**	0.000	0.780	**1.000**	0.000	0.000	**0.180**	0.000	0.000	0.000	0.000	**1.000**	**1.000**	**1.000**	0.140
10^{-5}	0.000	0.000	0.000	**1.000**	0.000	0.000	**0.180**	0.000	0.000	0.000	0.000	**1.000**	**1.000**	0.660	0.020
	F11			F12			F13			F14			F15		
ϵ	NMMSO	N-VMO	dADE	NMMSO	N-VMO	dADE	NMMSO	N-VMO	dADE	NMMSO	N-VMO	dADE	NMMSO	N-VMO	dADE
10^{-1}	**1.000**	**1.000**	0.640	**0.980**	0.220	**0.980**	0.960	**1.000**	0.140	0.080	**1.000**	0.700	0.000	**1.000**	**1.000**
10^{-2}	**1.000**	0.000	0.000	**0.980**	0.000	0.440	**0.960**	0.000	0.000	**0.060**	0.000	0.000	0.000	**0.020**	0.000
10^{-3}	**1.000**	0.000	0.000	**0.980**	0.000	0.000	**0.940**	0.000	0.000	**0.020**	0.000	0.000	0.000	0.000	0.000
10^{-4}	**1.000**	0.000	0.000	**0.980**	0.000	0.000	**0.940**	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
10^{-5}	**1.000**	0.000	0.000	**0.980**	0.000	0.000	**0.940**	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
	F16			F17			F18			F19			F20		
ϵ	NMMSO	N-VMO	dADE	NMMSO	N-VMO	dADE	NMMSO	N-VMO	dADE	NMMSO	N-VMO	dADE	NMMSO	N-VMO	dADE
10^{-1}	0.000	**1.000**	0.540	0.000	**1.000**	0.760	0.000	**0.960**	0.080	0.000	**0.220**	0.000	0.000	0.000	0.000
10^{-2}	0.000	**0.020**	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
10^{-3}	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
10^{-4}	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
10^{-5}	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

peaks found across runs, i.e. for q runs:

$$PR = \frac{\sum_{i=1}^{q} o_i}{tq} \tag{4}$$

where o_i denotes the number of global optima discovered by the ith run, and t is the total number of global peaks. We use the code made available by the CEC 2013 competition organisers for representing the test problems, and for assessing algorithm performance.[3]

We set the automatic merging tolerance $tol = 10^{-6}$, the maximum swarm size $n = 10D$ (where D is the number of design parameters) and the maximum number of swarms to increment $max\_inc = 100$. We use standard PSO parameters $c_1, c_2 = 2.0, \chi = 1.0$ but with a low inertia to promote local convergence ($\omega = 0.1$).

Table II shows the convergence rates of NMMSO and dADE for $\epsilon = 10^{-1}$, and these plus N-VMO for $\epsilon = 10^{-4}$ (convergence results for $\epsilon = 10^{-1}$ are not reported in [9]). At the $\epsilon = 10^{-1}$ level dADE performs the best, having the fastest mean convergence 10 times to NMMSO's nine. At the $\epsilon = 10^{-4}$ level however NMMSO performs much better than both the other algorithms, having faster convergence at this level on 12 of the problems. On the other eight problems none of the algorithms converged to all the global optima on any run at $\epsilon = 10^{-4}$.

Tables III and IV give the success rate and peak ratio results across the 20 test problems for all five levels of accuracy for each of the algorithms. NMMSO has the best/equal best SR

[3]Obtainable from http://goanna.cs.rmit.edu.au/~xiaodong/cec13-niching/competition/.

TABLE IV
MEAN PEAK RATIOS OF NMMSO, N-VMO AND DADE/NRAND/1/BIN. BEST VALUES FOR EACH PROBLEM AND ACCURACY LEVEL ARE UNDERLINED.

	F1			F2			F3			F4			F5		
ϵ	NMMSO	N-VMO	dADE	NMMSO	N-VMO	dADE	NMMSO	N-VMO	dADE	NMMSO	N-VMO	dADE	NMMSO	N-VMO	dADE
10^{-1}	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
10^{-2}	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
10^{-3}	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
10^{-4}	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
10^{-5}	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
	F6			F7			F8			F9			F10		
ϵ	NMMSO	N-VMO	dADE	NMMSO	N-VMO	dADE	NMMSO	N-VMO	dADE	NMMSO	N-VMO	dADE	NMMSO	N-VMO	dADE
10^{-1}	0.998	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**0.984**	0.412	0.837	0.930	**1.000**	**1.000**	**1.000**	**1.000**	0.985
10^{-2}	0.998	**1.000**	**1.000**	**1.000**	**1.000**	0.962	**0.984**	0.294	0.595	0.922	0.683	**1.000**	**1.000**	**1.000**	0.978
10^{-3}	0.998	0.940	**1.000**	**1.000**	0.945	0.892	**0.983**	0.270	0.545	0.920	0.399	**1.000**	**1.000**	**1.000**	0.981
10^{-4}	**0.997**	0.670	0.984	**1.000**	0.901	0.823	**0.981**	0.198	0.431	0.917	0.275	**1.000**	**1.000**	**1.000**	0.967
10^{-5}	0.000	0.000	0.000	**1.000**	0.806	0.732	**0.980**	0.027	0.356	0.913	0.192	**1.000**	**1.000**	0.968	0.947
	F11			F12			F13			F14			F15		
ϵ	NMMSO	N-VMO	dADE	NMMSO	N-VMO	dADE	NMMSO	N-VMO	dADE	NMMSO	N-VMO	dADE	NMMSO	N-VMO	dADE
10^{-1}	**1.000**	**1.000**	0.893	**0.998**	0.848	**0.998**	0.993	**1.000**	0.743	0.770	**1.000**	0.923	0.673	**1.000**	**1.000**
10^{-2}	**1.000**	0.667	0.667	**0.998**	0.745	0.887	**0.993**	0.667	0.667	**0.740**	0.667	0.667	0.673	**0.713**	0.620
10^{-3}	**1.000**	0.667	0.667	**0.998**	0.725	0.745	**0.990**	0.667	0.667	**0.713**	0.667	0.655	**0.673**	0.668	0.615
10^{-4}	**1.000**	0.667	0.667	**0.998**	0.713	0.740	**0.990**	0.667	0.667	**0.710**	0.667	0.655	**0.670**	0.623	0.627
10^{-5}	**1.000**	0.667	0.667	**0.998**	0.565	0.728	**0.990**	0.663	0.667	**0.703**	0.637	0.655	**0.668**	0.390	0.620
	F16			F17			F18			F19			F20		
ϵ	NMMSO	N-VMO	dADE	NMMSO	N-VMO	dADE	NMMSO	N-VMO	dADE	NMMSO	N-VMO	dADE	NMMSO	N-VMO	dADE
10^{-1}	0.663	**1.000**	0.873	0.553	**1.000**	0.938	0.633	**0.987**	0.683	**0.477**	0.340	0.420	**0.183**	0.000	0.030
10^{-2}	0.660	**0.703**	0.667	**0.548**	0.475	0.472	0.633	0.483	**0.660**	**0.470**	0.133	0.143	**0.180**	0.000	0.000
10^{-3}	0.660	0.653	**0.667**	**0.543**	0.440	0.417	**0.633**	0.470	0.630	**0.463**	0.133	0.063	**0.178**	0.000	0.002
10^{-4}	0.660	0.653	**0.667**	**0.538**	0.413	0.403	**0.633**	0.470	**0.633**	**0.447**	0.130	0.018	**0.178**	0.000	0.005
10^{-5}	0.660	0.633	**0.667**	**0.538**	0.320	0.410	**0.633**	0.360	0.627	**0.443**	0.103	0.000	**0.178**	0.000	0.000

value on 57% of the problem/accuracy level combinations. N-VMO achieves this for 44% and dADE 36%.[4] For 26% of the problem/accuracy level combinations *all* of the optimisers have an SR of zero. On the mean PR measure, NMMSO has the best/equal best value on 79% of the problem/accuracy level combinations. N-VMO achieves this for 43% and dADE 41%. No algorithm found any solutions at the $\epsilon = 10^{-5}$ level on F6. NMMSO tends to have relatively better performance at the higher accuracy levels. This would indicate that once NMMSO has identified a peak, the low inertia swarms are more effective at exploiting it to a high level of accuracy in a rapid fashion compared to the other algorithms.

Table V presents the overall performance assessment of NMMSO. The mean PR for the five accuracy levels on each of the 20 test functions are combining into a single average, and compared to the results of the 15 state-of-the-art entrants to the CEC 2013 competition (results from [1]), along with MEA and MSSPSO. NMMSO can be seen to be extremely competitive with the current state-of-the-art, having the highest mean and median PR yet observed.

Figure 2 shows the number of swarms maintained by each run of NMMSO for each of the problems. As can be seen, when there are only global modes, the number of swarms maintained converges to the number of global modes (F2, F4, F7, F9 and F10). For problems with very many local optima the number of swarms can be seen to constantly rise, however, due to the search bias toward the better performing mode estimates, this does not prevent good convergence results on these problems. Interestingly, on the composite functions with 10-20D NMMSO can be seen to consistently hone just a few peaks until at some point the number of swarms rises quickly.

[4]There appear to be some data entry issues in the tabulated results for the some of peak ratios in [8], as the values reported *increase* from $\epsilon = 10^{-3}$ to 10^{-4} for F15 and F18, and from $\epsilon = 10^{-2}$ to 10^{-4} for F20.

TABLE V
AVERAGE RESULTS ACROSS ALL TEST PROBLEMS AND ACCURACY LEVELS OF NMMSO, MEA AND MSSPSO, ALONG WITH RESULTS OF MULTI-MODAL ALGORITHMS COMPARED IN THE CEC 2013 COMPETITION (DETAILED IN [1]) ON THE PEAK RATIO.

Algorithm	Median	Mean	St. D.
NMMSO	**0.9933**	**0.8271**	0.2535
MEA [16]	0.2167	0.3676	0.3878
MSSPSO [15]	0.0000	0.2179	0.3901
A-NSGA-II [17]	0.0740	0.3275	0.4044
CMA-ES [7]	0.7750	0.7137	0.2807
CrowdingDE [18]	0.6667	0.5731	0.3612
dADE/nrand/1 [8]	0.7488	0.7383	0.3010
dADE/nrand/2 [8]	0.7150	0.6931	0.3174
DECG [19]	0.6567	0.5516	0.3992
DELG [19]	0.6667	0.5706	0.3925
DELS-aj [19]	0.6667	0.5760	0.3857
DE/nrand/1 [20]	0.6386	0.5809	0.3338
DE/nrand/2 [20]	0.6667	0.6082	0.3130
IPOP-CMA-ES [21]	0.2600	0.3625	0.3117
NEA1 [6]	0.6496	0.6117	0.3280
NEA2 [6]	0.8513	0.7940	0.2332
N-VMO [9]	0.7140	0.6983	0.3307
PNA-NSGA-II [22]	0.6660	0.6141	0.3421

This is probably due to tendency of NMMSO to merge local modes that live on larger landscape features. Figure 3 shows the growth of a swarm population over time as visualised in $\mathcal{X}$ for some of the 2D test problems. Where the global/local peaks/basins are deformations from the plane the number of modes identified quickly increases (see e.g. F6, and regions in F12 and F13), as the mid-points always tend to be lower between any pairing of global/local mode estimate. Where however local modes lie on a larger landscape feature (e.g. the peak mid way down on the right of F11 and F12), these swarms tend not to be sustained, as pairing with a swarm converging on a mode 'further' up the larger landscape feature

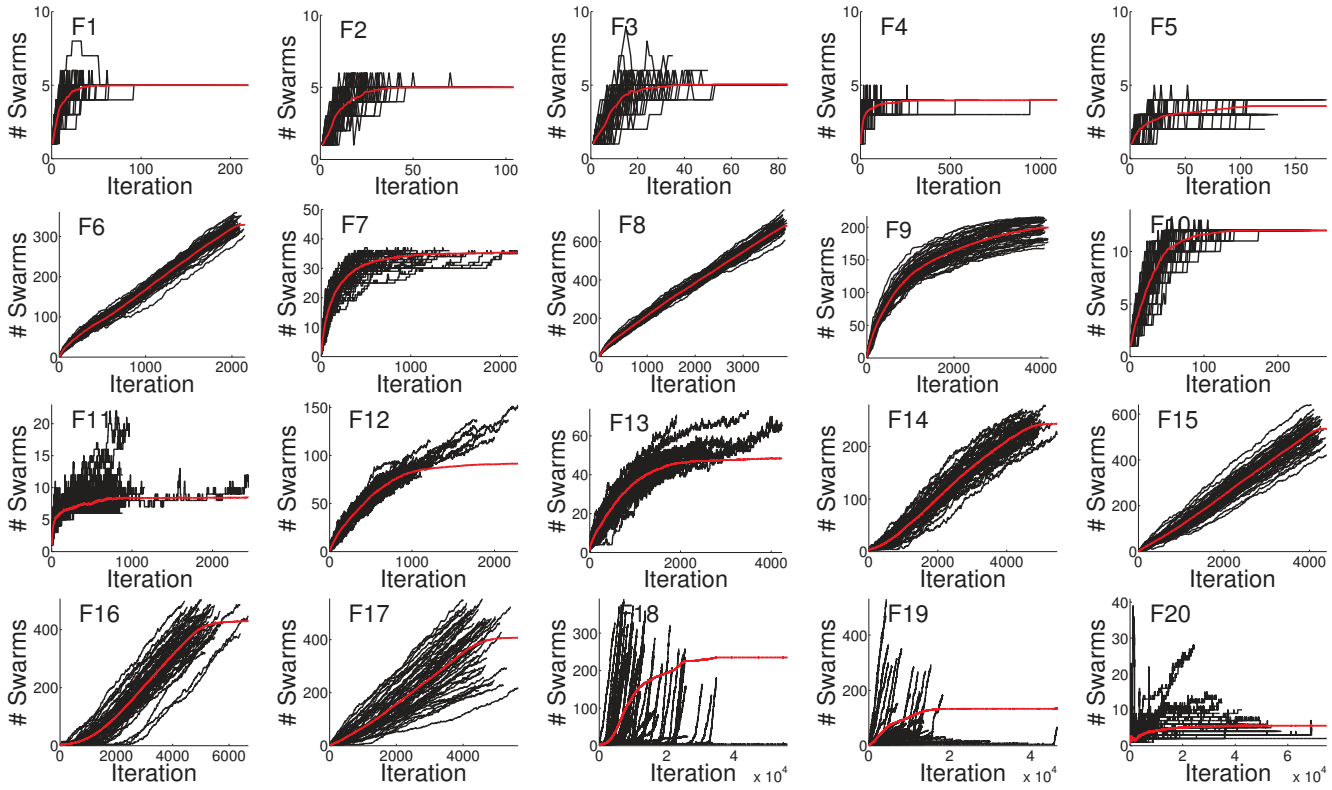

Fig. 2. Number of swarms maintained by each run on each problem, recorded at each iteration until all global optima have converged to within 10^{-5}, or the function evaluations allowed are exhausted. Swarms recorded at line 11 of Figure 1. Mean of runs plotted in red (when a run has terminated, the size of its final population is used in the calculation of the mean swarm size until all runs complete).

will tend to result in a mid-point that is higher than the lower swarm gbest – causing the paired swarms to be merged.

VI. Discussion

We have introduced a new multi-modal optimiser which utilises a number of self-contained but communicating sub-swarms to search for modes in the fitness landscape. The approach builds on a number of properties which have been identified in effective multi-modal optimisers, and continuously hones mode estimates by exploiting individual swarms in parallel, rather than searching in local regions and moving on like other multi-swarm approaches to multi-modal optimisation. Particles may additionally split off to form new swarms, or migrate between swarms by splitting and subsequently merging. Results on the CEC 2013 niching test problems show that the proposed algorithm is extremely competitive with the current state-of-the-art, and gives robust performance, even though seeded by a single random solution in $\mathcal{X}$. Nevertheless, the number of parameters is still larger than would be generally desirable: future work will be focused on reducing these (via self-adaptation) as well as applying NMMSO to multi-modal engineering design problems. Note however that the optimiser does not require a niching radius or number of niches to be set – as it dynamically fits both of these locally, based upon the distance between the current peak estimate locations being maintained. MATLAB code for the NMMSO, MEA and MSSPSO algorithms is available at https://github.com/fieldsend.

Acknowledgement

The author would like to thank Prof. Xiaodong Li, Prof. Andries Engelbrecht, and Dr Michael Epitropakis, the organisers of the CEC 2013 "Competition on Niching Methods for Multimodal Optimization", for making the competition test problem code and evaluation functions readily available online for the community.

References

[1] X. Li, A. Engelbrecht, and M. Epitropakis, "Results of the 2013 IEEE CEC Competition on Niching Methods for Multimodal Optimization." Presented at 2013 IEEE Congress on Evolutionary Computation Competition on: Niching Methods for Multimodal Optimization, 2013.

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

[3] 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*, 1987, pp. 41–49.

[4] B. Sareni and L. Krähenbühl, "Fitness Sharing and Niching Methods Revisited," *IEEE Transactions on Evolutionary Computation*, vol. 2, no. 3, pp. 97–106, 1998.

[5] 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.

[6] M. Preuss, "Niching the CMA-ES via Nearest-better Clustering," in *Proceedings of the 12th Annual Conference Companion on Genetic and Evolutionary Computation*, ser. GECCO '10, 2010, pp. 1711–1718.

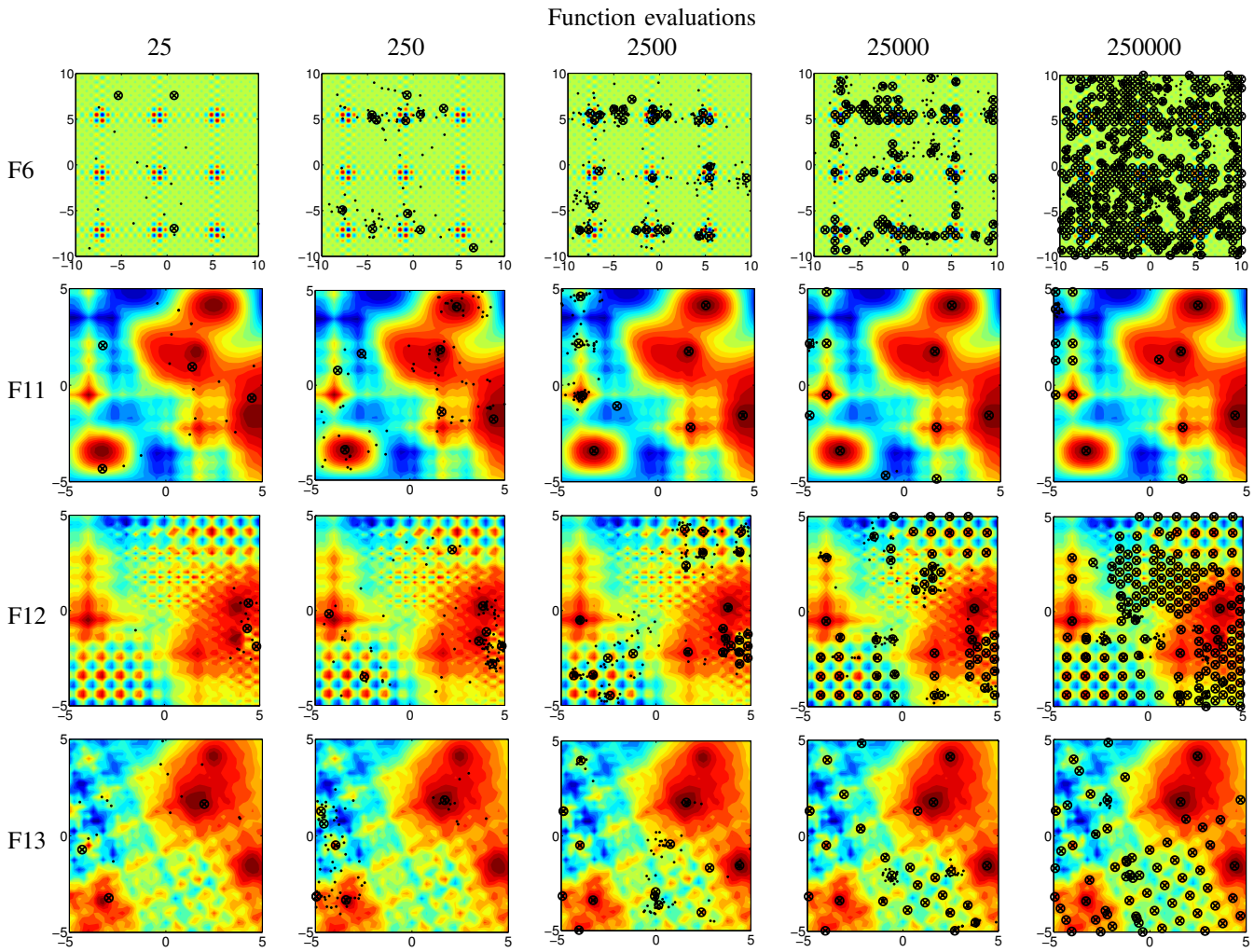

Fig. 3. Swarm gbests (marked with a '⊗'), and swarm particles (marked with a '·') of NMMSO at different numbers of function evaluations for five of the 2D test problems. Function response coloured from high (dark red) to low (dark blue).

[7] N. Hansen and A. Ostermeier, "Completely Derandomized Self-Adaptation in Evolution Strategies," *Evolutionary Computation*, vol. 9, no. 2, pp. 159–195, 2001.

[8] M. G. Epitropakis, X. Li, and E. K. Burke, "A dynamic archive niching differential evolution algorithm for multimodal optimization," in *IEEE Congress on Evolutionary Computation*, 2013, pp. 79–86.

[9] D. Molina, A. Puris, R. Bello, and F. Herrera, "Variable mesh optimization for the 2013 CEC Special Session Niching Methods for Multimodal Optimization," in *IEEE Congress on Evolutionary Computation*, 2013, pp. 87–94.

[10] J. E. Fieldsend, "Multi-Modal Optimisation using a Localised Surrogates Assisted Evolutionary Algorithm," in *UK Workshop on Computational Intelligence (UKCI 2013)*, 2013, pp. 88–95.

[11] ——, "Using an adaptive collection of local evolutionary algorithms for multi-modal problems," *Soft Computing*, in press.

[12] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *IEEE International Conference on Neural Networks*. Perth, Australia: IEEE Service Center, 1995, pp. 1942–1948.

[13] J. Fieldsend, "Multi-objective particle swarm optimisation methods," Department of Computer Science, University of Exeter, Tech. Rep. 419, March 2004.

[14] S. Chen, "Locust Swarms - A new multi-optima search technique," in *IEEE Congress on Evolutionary Computation*, 2009, pp. 1745–1752.

[15] J. Zhang, D.-S. Huang, and K.-H. Liu, "Multi-Sub-Swarm Optimization Algorithm for Multimodal Function Optimization," in *IEEE Congress on Evolutionary Computation*, 2007, pp. 3215–3220.

[16] R. K. Ursem, "Multinational evolutionary algorithms," in *Proceedings of the Congress on Evolutionary Computation*, 1999, pp. 1633–1640.

[17] K. Deb and A. Saha, "Multimodal Optimization Using a Bi-objective Evolutionary Algorithm," *Evolutionary Computation*, vol. 20, no. 1, pp. 27–62, 2012.

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

[19] J. Ronkkonen, "Continuous Multimodal Global Optimization with Differential Evolution-Based Methods," Ph.D. dissertation, Lappeenranta University of Technology, Finland, 2009.

[20] M. G. Epitropakis, V. P. Plagianakos, and M. N. Vrahatis, "Finding multiple global optima exploiting differential evolution's niching capability," in *IEEE Symposium on Differential Evolution (SDE)*, 2011, pp. 1–8.

[21] A. Auger and N. Hansen, "A restart CMA evolution strategy with increasing population size," in *IEEE Congress on Evolutionary Computation*, vol. 2, 2005, pp. 1769–1776.

[22] S. Bandaru and K. Deb, "A parameterless-niching-assisted bi-objective approach to multimodal optimization," in *IEEE Congress on Evolutionary Computation*, 2013, pp. 95–102.

[23] X. Li, A. Engelbrecht, and M. G. 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.