2014 IEEE Congress on Evolutionary Computation (CEC)
July 6-11, 2014, Beijing, China

Extending Minimum Population Search towards Large Scale Global Optimization

Antonio Bolufé-Röhler and Stephen Chen

***Abstract*—Minimum Population Search is a new metaheuristic specifically designed for optimizing multi-modal problems. Its core idea is to guarantee exploration in all dimensions of the search space with the smallest possible population. A small population increases the chances of convergence and the efficient use of function evaluations – an important consideration when scaling a search technique up towards large scale global optimization. As the cost to converge to any local optimum increases in high dimensional search spaces, metaheuristics must focus more and more on gradient exploitation. To successfully maintain its balance between exploration and exploitation, Minimum Population Search uses thresheld convergence. Thresheld convergence can ensure that a search technique will perform a broad, unbiased exploration at the beginning and also have enough function evaluations allocated for proper convergence at the end. Experimental results show that Minimum Population Search outperforms Differential Evolution and Particle Swarm Optimization on complex multi-modal fitness functions across a broad range of problem sizes.**

I. Introduction

Large scale global optimization (LSGO) has been recently arising in many real-world applications such as bio-computing [1], telecommunication [2], and financial markets [3]. With increasing dimensionality, the search space volume grows exponentially. Known as "the curse of dimensionality" [4], the exponential increase of candidate solutions leads to a decrease in the performance of most optimization algorithms [5].

Although the search space increases exponentially, the computational resources available often increase only linearly with respect to the dimensions (d) of the problem. For example, in one large scale global optimization contest (i.e. $d \geq 1000$), the allowed function evaluations (FEs) was FEs = 3000*d [5]. These mismatched growth rates between search space volume and available FEs can cause adequate search space coverage in low dimensional search spaces to become exceptionally sparse coverage in high dimensional search spaces. As a consequence, heuristics in large scale global optimization need to become highly efficient in the use of the available function evaluations.

As the cost to converge to any local optimum increases, the ability to dedicate any effort to exploration decreases rapidly. In [6], it is argued that in high dimensional search spaces, metaheuristics must focus almost exclusively on gradient exploitation. Favoring exploitation over exploration has important consequences when optimizing multi-modal problems. In multi-modal search spaces, an extra amount of FEs has to be put into exploration in order to avoid/escape poor local optima. Finding the correct balance between exploration and exploitation becomes critical in large scale global optimization. Exploring too much may not leave enough function evaluations to converge, but dedicating all of the computational resources to gradient exploitation can also negatively affect performance on multi-modal functions.

Minimum Population Search (MPS) [7] is designed to guarantee full coverage of a search space with a small population. It is hoped that its efficient use of function evaluations will allow MPS to scale well to high dimensional search spaces. MPS has been shown to have competitive performance on multi-modal problems, especially with low budgets of FEs [7]. The current work presents a first step towards extending MPS towards large scale global optimization. As part of this extension, a new adaptive version of thresheld convergence has been developed. In MPS, the role of thresheld convergence is to control the transition from exploration to exploitation, and increasing dimensionality offers new opportunities and challenges in this important task.

The next section presents a background on large scale global optimization. Section III then describes the development of MPS. Computational results for MPS and other metaheuristics are presented for a broad range of problem sizes in Section IV. The new adaptive version of thresheld convergence for MPS is presented in Section V. Finally, a discussion about the new metaheuristic is carried out in Section VI before the paper is summarized in Section VII.

II. Background

Previous work on large scale global optimization can be roughly classified into two main approaches: optimizing the problems as wholes or decomposing them into a number of sub-problems. Partitioning a large problem into sub-problems and then solving them independently is a very intuitive methodology. In recent years, several decomposition-based algorithms have been proposed, ranging from total decomposition (solving d 1-dimensional problems) [8] to capturing the interacting variables and grouping them in one subcomponent [9, 10].

Decomposition-based algorithms, however, may not be as effective when facing non-separable problems [11]. In non-separable problems, a proportion of the decision variables have interactions amongst themselves. In such

A. Bolufé-Röhler is with the University of Havana, Havana, Cuba (phone: 53-5-242-0064; e-mail: bolufe@matcom.uh.cu).

S. Chen is with York University, Toronto, ON, Canada (e-mail: sychen@yorku.ca).

cases, it becomes necessary to detect the interacting variables and to group them in the same sub-problem. Unfortunately, detecting correlations may be computational costly, the clusters of interacting variables may be large enough to consider the decomposed sub-problem as a large scale problem by itself, and/or the objective function may not be separable at all [5].

When optimizing non-separable problems, increasing the scalability of a specific operator or a general search strategy is a complementary approach. Previous works focus on modifying the classical operators (which are usually developed for low dimension tasks) in an attempt to boost their performance in high dimensions [12]. Techniques such as self-adaptation, hybridization, and population reduction have been used to increase the scalability of different heuristic methods [12−14]. The Minimum Population Search (MPS) metaheuristic falls into this general group. However, instead of attempting to improve scalability from existing methods, the search strategy of MPS was designed from the very beginning to be scalable. The main idea is to provide strong exploration mechanisms which allow it to effectively cover the search space while using a (relatively) small population [7].

Smaller populations allow more generations and increase the efficient use of FEs. However, if the population size (n) becomes smaller than the dimensionality of the problem (d) then its population will define an n-1 dimensional hyperplane. New solutions generated strictly from the line segments formed among the population members (e.g. difference vectors, attraction vectors, mid-point crossover, etc) will "get trapped" inside this n-1 dimensional hyperplane which is a subset of the complete search space. This is an important consideration since line segments are an important feature of many optimization techniques such as Nelder-Mead (NM) [15], Differential Evolution (DE) [16], and Particle Swarm Optimization (PSO) [17].

MPS has been specifically designed to deal with this limitation. Using a population size equal to the dimensionality of the problem ($n = d$), new solutions are generated using difference vectors to be in a d-1 dimensional hyperplane. Full coverage of the search space is then achieved by taking a subsequent step that is orthogonal to this hyperplane. To preserve diversity and avoid premature convergence, the size of the hyperplane and orthogonal steps is controlled using thresheld convergence [18, 19]. The minimum step (threshold) decays as the search progresses and convergence is thus "held" back until the last stages of the search process. By controlling the decay rate of the threshold function, it is possible to effectively determine the amount of exploration and exploitation performed by the algorithm.

III. MINIMUM POPULATION SEARCH

Minimum Population Search is a recently developed metaheuristic specifically designed for optimizing multi-modal problems. Another distinctive feature of MPS is that its search strategy was explicitly designed from the beginning to be scalable to large scale global optimization. The key ideas were initially developed for two dimensional problems in [7] and later generalized for standard dimensions in [20].

A. Minimum Population Search for Two Dimensions

MPS can use the minimum population size of $n = 2$ (as a population size of 1 makes a population-based technique indistinguishable from a point-search technique). In two dimensions, each iteration of MPS starts with the generation of "line points" along the subspace (line) determined by the two population members (x_1 and x_2). The "line points" are generated by adding the (normalized) difference vector formed by x_1 and x_2 to each population member x_i. The direction and size of the difference vector is determined by the scaling factor F_i (1).

$$line_i = x_i + F_i * (x_1 - x_2) \quad (1)$$

Taking a step along the line segment formed by x_1 - x_2 will only generate solutions in the subspace (line) defined by the two population members. An orthogonal step to this subspace (line) allows the search process of MPS to cover the full dimensionality of the (2D) problem. The direction and size of this exploratory step is determined by the O_{step_i} factor (2).

$$trial_i = line_i + O_{step\_i} * orth \quad (2)$$

Thresheld convergence forces new solutions to be a minimum (min_step) threshold distance away from their parent solutions. To guarantee that the distance between a "line point" and its parent solution is smaller than the maximum allowed step (max_step), F_i is drawn with a uniform distribution from the [$-max\_step$, max_step] interval (note: the x_1-x_2 vector is normalized before scaling). To

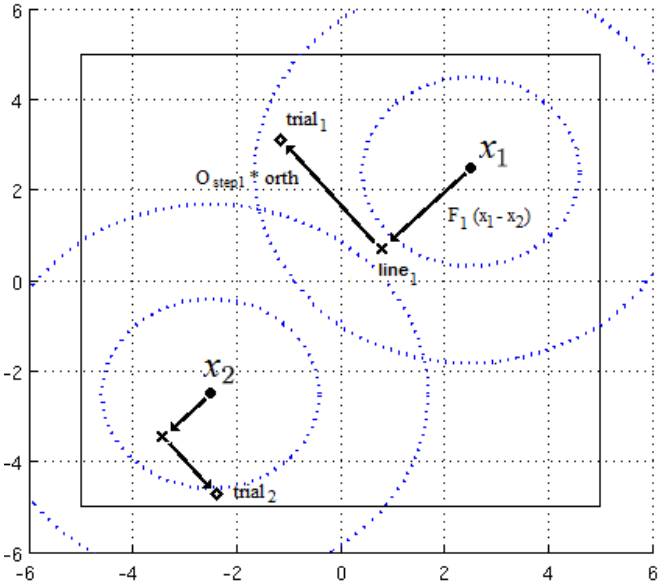

Fig. 1. Visualization of MPS search process in 2D: x_1 and x_2 are the current population members, the crosses are the "line points" formed from (1) and the diamonds are the trial points that result after (2). The dotted circle lines show the minimum and maximum step thresholds around x_1 and x_2.

ensure that the distance from the new trial solution ($trial_i$) to its parent solution (x_i) stays within the acceptable [min_step, max_step] threshold range, the O_{step_i} factor is selected with a uniform distribution from the [min_orth_i, max_orth_i] interval (note: the *orth* vector is normalized before scaling). The min_orth_i and max_orth_i values are calculated by (3) and (4), respectively. Since the x_1-x_2 vector is normalized before scaling, the F_i factor represents the actual distance between the "line point" ($line_i$) and its corresponding parent solution (x_i). New solutions which fall outside the feasible search space are clamped back to the boundaries. The two best solutions from the current population and the new trial solutions survive into the next generation.

$$min\_orth_i = \sqrt{\max(\min\_step_i^2 - F_i^2, 0)} \quad (3)$$

$$max\_orth_i = \sqrt{\max(\max\_step_i^2 - F_i^2, 0)} \quad (4)$$

The min_step and max_step values are updated by a rule similar to that used in previous attempts to control convergence for DE [18] and PSO [19] in which an initial threshold is selected that then decays over the course of the search process, see (5). Equation (5) shows how min_step is calculated (note: $max\_step = 2 * min\_step$). In (5), α represents a fraction of the main space diagonal, FEs is the total available amount of function evaluations, k is the number of evaluations used so far, and γ is the parameter that controls the decay rate of the threshold. The implementation presented in [7] uses $\alpha = 0.3$ and $\gamma = 3$.

$$min\_step_i = \alpha * diagonal * ([FEs - k]/FEs)^{\gamma} \quad (5)$$

To ensure good spacing in the initial population, the initial points are selected to be on the diagonal of the search space. Assuming that the search space is bounded by the same lower and upper bound in each dimension (as in the used benchmark functions), the initial points are selected as x_1=*(bound/2, bound/2)* and x_2 =*(-bound/2, -bound/2)*. Fig. 1 shows the search strategy of MPS in a two-dimensional search space.

B. MPS in Standard Dimensions

If the population size becomes smaller than dimensionality of the problem, line segments will be restricted to the *n-1* hyperplane formed by the *n* population members. To avoid new solutions from getting trapped in this subspace of the whole search space, the population of MPS increases with dimensionality (i.e. $n = d$). The orthogonal step can then guarantee searching into all *d* dimensions of the search space. This full extension of MPS from the two dimensional version would require adding *d-1* difference vectors to the parent solution and using a vector orthogonal to these *d-1* difference vectors as the orthogonal step. A difficulty of this approach is the high computational cost of calculating the orthogonal vector. Thus, a version of MPS which is conceptually simpler and has a lower computational cost was developed based on the centroid of the population [20].

In MPS-centroid (or simply MPS), the "hyperplane points" are obtained by adding to the parent solution the difference vector between the parent and the centroid (instead of *d-1* difference vectors). The orthogonal step is done using a vector orthogonal to the parent-centroid line (2D subspace). Fig. 2 shows a graphical comparison in 3D between the MPS version using a vector orthogonal to the *d-1* hyperplane and the centroid version. In the centroid version, the offspring solution (*trial*) can be on an entire plane (perpendicular to the centroid vector) as opposed to only a line that is perpendicular to the plane defined by x_1, x_2, and x_3.

In higher dimensions each population member is initialized using (6): s_k is the k^{th} population member, rs_i are random numbers which can be -1 or 1, and *bound* is the lower and upper bound on each dimension. This initialization method leads to a better distribution of the initial solutions in the search space than did uniform random solutions.

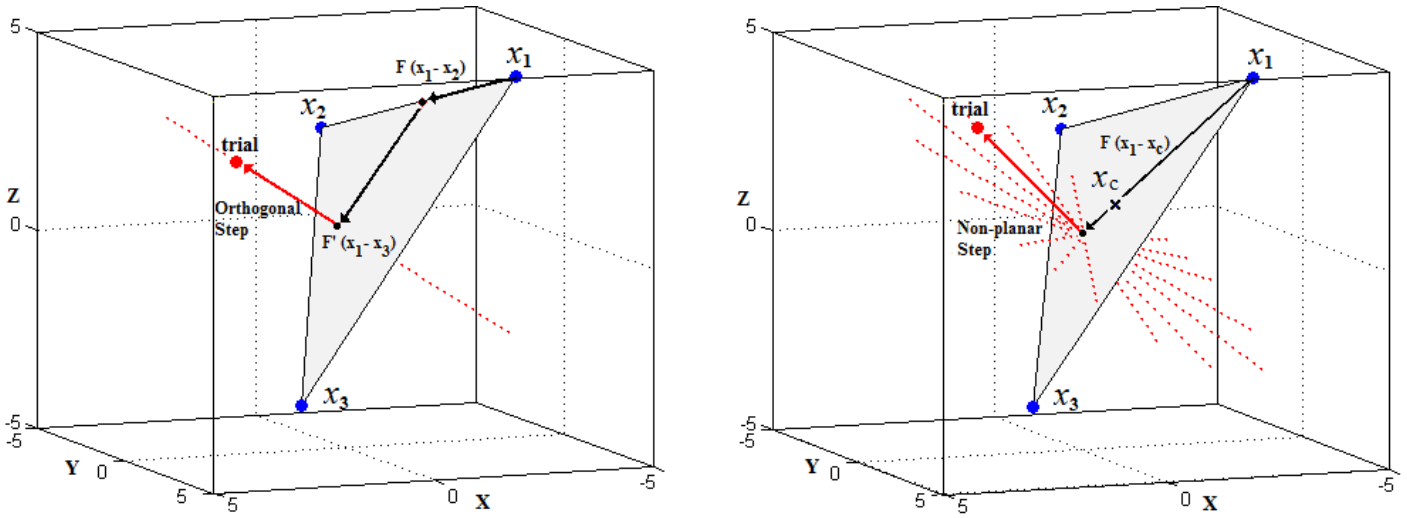

Fig. 2. An illustrated comparison between using two difference vectors (left) and the centroid (right) for generating a new trial point in MPS. In both cases x_1, x_2, and x_3 are the (parent) solutions from the current population.

Algorithm 1 Minimum Population Search

```
MPS (α, γ, maxFEs)
X ← InitialPopulation()                                  // Equation (6)
while FEs < maxFEs
    min_step ← UpdateThreshold(α, γ)                     // Equation (5)
    max_step ← min_step*2
    x_c ← CalculateCentroid()
    for i = 1 : popsize
        F_i ← UniformRandom(-max_step, max_step)
        orth_i ← OrthogonalVector(x_i - x_c)             // normalized vector
        orth_step ← UniformRandom(min_orth, max_orth)    // Equations (3) and (4)
        trial_i ← x_i + F_i*(x_i - x_c) + orth_step*orth_i   // clamping if necessary
    endfor
    X ← BestSolutions(X, trial)
endwhile
```

$$s_k = (rs_i * bound/2,\ rs_2 * bound/2,\ \ldots,\ rs_n * bound/2) \quad (6)$$

At each generation a simple set of operations are performed. First, the threshold values are updated (5) and the centroid is calculated. Then, each member is used as a parent solution to generate an offspring. The mechanism used to generate the new solutions is similar to the two-dimensional MPS, but the centroid is used instead of the other population member. The "hyperplane points" are obtained by adding the parent-centroid difference vector to the parent solution. The orthogonal step is made taking a random vector orthogonal (*orth*) to the parent-centroid difference vector. This two-step process for generating the new trial solutions ($trial_i$) is presented in (7).

$$trial_i = x_i + F_i * (x_i - x_c) + O_{step\_i} * orth \quad (7)$$

In (7), x_i and x_c are the parent and the centroid, respectively. The F_i factor is drawn with a uniform distribution from [*-max_step, max_step*] (x_i-x_c is normalized before scaling). The O_{step_i} factor is selected with a uniform distribution from [min_orth_i, max_orth_i] (the *orth* vector is also normalized). The min_orth_i and max_orth_i values are calculated as in the two-dimensional version using (3) and (4). Once the new solutions are created, clamping is performed if necessary, and the best *n* solutions among the parents and offspring survive into the next generation. The parameters for the threshold function are $\alpha=0.3$ and $\gamma=3$. A detailed pseudo-code is presented in Algorithm 1.

IV. Computational Results

A set of experiments has been designed to test the effectiveness of MPS. The experiments have been performed using the Black-Box Optimization Benchmark (BBOB) minimization functions [21] with a budget of FEs = 3000*d, as in large scale global optimization contests [5]. There are 24 BBOB functions divided into five sets. Since MPS is explicitly designed for multi-modal search spaces, this paper focuses on Set 4-multi-modal functions with adequate global structure and Set 5-multi-modal functions with weak global structure. In Table I, the names of these functions are indicated. The experiments include comparisons to PSO and DE. The PSO algorithm is a standard version with ring topology [17], zero initial velocities [22], and "Reflect-Z" for particles that exceed the boundaries of the search space [23]. The DE method is the highly common and frequently effective variant labeled DE/rand/1/bin [16].

TABLE I
BBOB Functions

Set		Function Name	Attribute		
			s	u	gs
1	1	Sphere	X	X	X
	2	Ellipsoidal, original	X	X	X
	3	Rastrigin	X		X
	4	Büche-Rastrigin	X		X
	5	Linear Slope	X	X	
2	6	Attractive Sector		X	
	7	Step Ellipsoidal			X
	8	Rosenbrock, original			
	9	Rosenbrock, rotated			
3	10	Ellipsoidal, rotated		X	X
	11	Discus		X	X
	12	Bent Cigar		X	
	13	Sharp Ridge		X	
	14	Different Powers		X	
4	15	Rastrigin, rotated			X
	16	Weierstrass			X
	17	Schaffers F7			X
	18	Schaffers F7, moderately ill-conditioned			X
	19	Composite Griewank-Rosenbrock F8F2			X
5	20	Schwefel			
	21	Gallagher's Gaussian 101-me Peaks			
	22	Gallagher's Gaussian 21-hi Peaks			
	23	Katsuura			
	24	Lunacek bi-Rastrigin			

Names and selected attributes of the 24 functions in the BBOB problem set – separable (s), unimodal (u), global structure (gs).

TABLE II
PERFORMANCE AS DIMENSIONS INCREASE (FES = 3000*d)

Set	F	MPS	DE n=50	PSO n=50	DE n=2*d	PSO n=2*d
			d = 5			
	15	**1.75e+00**	6.35e+00	3.16e+00	5.66e+00	4.12e+00
	16	**1.77e−01**	1.35e+00	1.87e−01	1.02e+00	3.93e−01
4	17	3.85e−04	**1.79e−05**	1.84e−02	8.86e−03	6.77e−02
	18	3.30e−02	**4.79e−04**	1.19e−01	1.91e−01	3.96e−01
	19	**2.23e−01**	7.16e−01	3.08e−01	1.08e+00	3.76e−01
	20	5.87e−01	2.66e−01	4.04e−01	**2.58e−01**	5.70e−01
	21	8.24e−01	**2.86e−01**	4.40e−02	1.03e+00	8.35e−01
5	22	9.87e−01	2.72e−01	**1.59e−01**	1.19e+00	9.82e−01
	23	**2.37e−01**	9.79e−01	5.36e−01	1.12e+00	5.71e−01
	24	**5.92e+00**	1.22e+01	7.85e+00	7.96e+00	8.00e+00
			d = 10			
	15	**5.02e+00**	3.32e+01	1.57e+01	1.99e+01	1.55e+01
	16	**5.60e−01**	8.19e+00	1.85e+00	3.84e+00	2.52e+00
4	17	1.07e−02	9.73e−04	2.74e−01	**1.87e−04**	2.03e−01
	18	1.95e−01	**2.72e−02**	8.95e−01	3.59e−02	1.17e+00
	19	**4.27e−01**	2.72e+00	1.81e+00	2.53e+00	1.63e+00
	20	1.22e+00	1.82e+00	8.38e−01	**5.02e−01**	8.77e−01
	21	2.23e+00	1.18e+00	**6.83e−01**	1.80e+00	9.00e−01
5	22	3.43e+00	1.82e+00	**1.40e+00**	2.95e+00	2.60e+00
	23	**1.49e−01**	1.47e+00	8.40e−01	1.42e+00	8.93e−01
	24	**1.39e+01**	4.50e+01	3.45e+01	3.87e+01	3.41e+01
			d = 20			
	15	**1.12e+01**	1.15e+02	6.13e+01	1.11e+02	4.15e+01
	16	**1.51e+00**	1.80e+01	6.72e+00	1.82e+01	5.11e+00
4	17	2.84e−02	5.67e−03	1.05e+00	**7.44e−04**	1.00e+00
	18	4.15e−01	1.91e−01	3.59e+00	**7.25e−02**	3.03e+00
	19	**5.67e−01**	4.93e+00	3.94e+00	4.95e+00	2.07e+00
	20	1.68e+00	2.57e+00	1.17e+00	2.10e+00	**1.08e+00**
	21	5.39e+00	4.67e+00	**8.64e−01**	4.22e+00	9.13e−01
5	22	7.04e+00	3.26e+00	**2.15e+00**	2.81e+00	2.24e+00
	23	**1.98e−01**	2.17e+00	1.47e+00	2.06e+00	1.15e+00
	24	**3.09e+01**	1.31e+02	1.21e+02	1.33e+02	8.79e+01
			d = 50			
	15	**5.19e+01**	3.87e+02	2.93e+02	3.40e+02	2.02e+02
	16	**3.91e+00**	3.26e+01	1.65e+01	2.27e+01	1.15e+01
4	17	1.31e−01	**1.54e−02**	2.93e+00	3.52e−01	2.07e+00
	18	1.58e+00	**3.15e−01**	1.22e+01	3.13e+00	1.02e+01
	19	**8.78e−01**	6.96e+00	7.06e+00	6.44e+00	5.24e+00
	20	2.24e+00	2.81e+00	**1.44e+00**	2.98e+00	1.53e+00
	21	4.46e+00	3.82e+00	**1.93e+00**	4.54e+00	1.57e+00
5	22	4.69e+00	6.00e+00	2.66e+00	4.37e+00	**2.57e+00**
	23	**2.50e−01**	3.66e+00	2.85e+00	3.12e+00	2.53e+00
	24	**8.38e+01**	4.37e+02	4.87e+02	4.01e+02	4.03e+02
			d = 100			
	15	**1.36e+02**	9.29e+02	8.16e+02	9.33e+02	1.06e+03
	16	**7.14e+00**	4.23e+01	2.50e+01	2.70e+01	2.53e+01
4	17	**3.02e−01**	3.12e−01	5.20e+00	2.12e+00	5.68e+00
	18	2.83e+00	**1.33e+00**	1.96e+01	6.12e+00	2.15e+01
	19	**1.52e+00**	7.96e+00	9.47e+00	7.10e+00	9.90e+00
	20	2.30e+00	**1.54e+00**	1.82e+00	4.42e+00	7.32e+00
	21	3.92e+00	6.25e+00	**2.02e+00**	5.05e+00	2.49e+00
5	22	**2.51e+00**	8.17e+00	3.05e+00	5.65e+00	2.82e+00
	23	**3.37e−01**	4.24e+00	3.63e+00	3.33e+00	3.46e+00
	24	**2.15e+02**	9.98e+02	1.25e+03	9.60e+02	1.33e+03

Mean error over 25 trials from known optimum.

A. Population Size

With increasing dimensionality, it seems natural to increase the population size accordingly. Despite the exponential growth of the search space, recommendations for population sizes include linear growth and constant size. A simple guideline from Differential Evolution is to use a population size that is ten times the dimensionality of the search space [24]. In standard PSO, a fixed population size of n = 50 is recommended [17].

The next set of experiments compares the performance of MPS, DE, and PSO, and it aims to show the relationship between population size and performance as dimensions increase. Table II presents the results for problem sizes of d = 5, 10, 20, 50, and 100. DE and PSO are tested with a constant population size (n = 50) and with a linearly increasing population size (n = 2*d). The presented results are the mean errors from optimum for five trials on each of the first five instances of each function (i.e. 25 independent trials for each function).

Results in Table II show that the relative performance of MPS versus DE and PSO increases as problem size increases. For d = 100, MPS provides the best results on 7 of the 10 functions. It can also be noticed that the DE and PSO versions with the increasing population size (n = 2*d) achieve the best performance around d = 20 and comparatively worse results in smaller and larger dimensions. To see these trends more clearly, Fig. 3 shows the average performance of these algorithms compared to MPS. The reported values are the average relative performances (%-diff = (a-b)/max(a,b)) achieved by MPS versus DE and PSO on BBOB Sets 4 and 5. These values indicate by what amount (percent) the given algorithm (b) outperforms MPS (a) – negative values indicate that the algorithm (DE or PSO) is outperformed by MPS.

Two key results are the steady decrease in performance of DE and PSO when compared against MPS, and how a population size correlated to the dimensionality d leads to poor performance for DE and PSO on small and large problems. When d and n are small, it appears that the population does not achieve the minimum required size for the search strategies of DE and PSO to become effective. When d and n are large, it appears that the large populations lead to an inefficient use of FEs (which prevents DE and PSO from converging). Since the best overall results are achieved with a constant population, the rest of the experiments of this paper use a population size of n = 50 for DE and PSO.

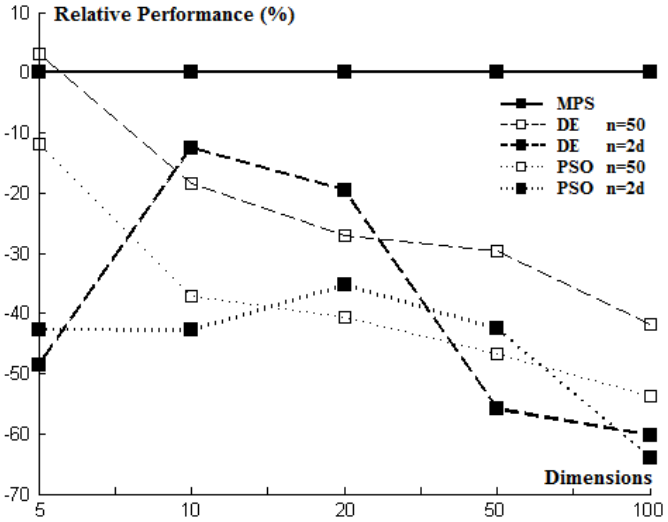

Fig. 3. Relative performance of MPS on BBOB Sets 4 and 5 versus DE and PSO.

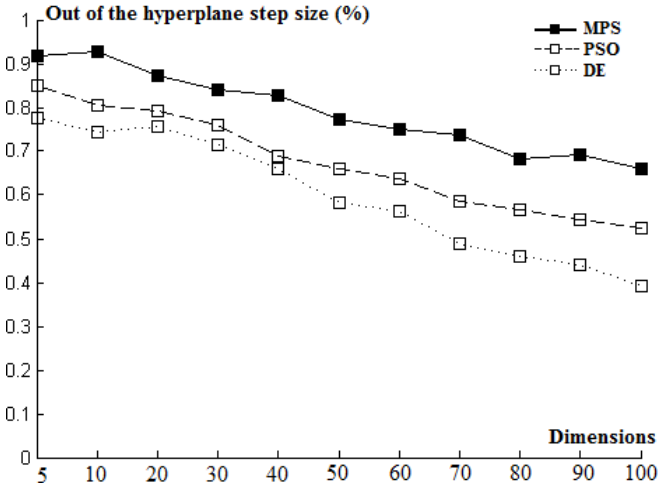

Fig. 4. Step size into the "missing dimension".

B. Hyperplane Search

If the population size of an evolutionary algorithm is smaller than the dimensionality of the problem then its population will define an n-1 dimensional hyperplane. New solutions generated strictly from the line segments formed among the population members will "get trapped" inside this n-1 dimensional hyperplane which is a subset of the complete search space. This situation occurs for DE and PSO where the primary mechanisms of difference vectors and attraction vectors act within the n-1 dimensional hyperplane defined by the n population members. To escape the n-1 dimensional hyperplane these heuristics rely on secondary search mechanisms. In PSO, the secondary mechanism is the "random vectors" of ε_1 and ε_2 which involve independent random numbers generated during each position update for each individual dimension. In DE, it is the use of a crossover operator which acts on the axial dimensions as opposed to the population's hyperplane.

The following experiment is designed to compare the effectiveness of the "out of the hyperplane" exploration of DE, PSO, and MPS. For the three algorithms, the population size was set equal to the dimensionality of the problem (i.e. $n = d$) and the distance from each new solution to the current d-1 hyperplane was measured. The average step size into this "missing dimension" was compared to the average step size for each dimension within the hyperplane. Fig. 4 shows the size of the step into the "missing dimension" compared to the steps in the "hyperplane dimensions" (as a percentage). Results are presented for d = 5-100 dimensions.

As it can be seen in Fig. 4, the ability of MPS to search out of the hyperplane can be almost similar to its ability to search inside the hyperplane. However, as dimensions increase the search steps beyond the hyperplane become smaller compared to the inside dimensions. The same happens for DE and PSO. In the case of DE, its ability to search beyond the hyperplane is clearly restricted. Although the effects of dimension-by- dimension crossover can allow escapes from the hyperplane, the small ratio of activity in these "exterior" dimensions means that DE will adjust quite slowly to new hyperplanes. PSO, shows an out of the hyperplane exploration in between DE and MPS.

V. An Adaptive Threshold Function for MPS

In large scale global optimization, it becomes necessary to "wisely exploit response surfaces and possess high efficiency" in the use of FEs [6]. Previous work has shown that MPS satisfies the second requirement. However, MPS does not directly exploit search gradients. Instead, it attempts to provide a full coverage of the search space by performing a methodical exploration based on controlled minimum and maximum search steps. Although this search strategy is effective for solving multi-modal problems, MPS may benefit from an increased exploitation when performing large scale global optimization. The threshold function is best suited for the task of balancing exploration and exploitation. By adjusting the threshold decay rate, it is possible to control the transition from exploring the search space and detecting the best regions to performing local search and exploiting the best regions already found.

Larger populations in higher dimensional problems provide information that MPS can use to develop adaptive threshold functions and improve performance [25]. However, determining the correct balance between exploration and exploitation does not only depend on the dimensionality of a problem, but on the characteristics of the function as well. If the distance among the different attraction basins could be known *a priori*, then the threshold could be held longer at this "ideal search scale". Allocating more function evaluations to this threshold level could promote a better exploration of different basins and increase the chances of finding the best regions before performing a more intense/local search.

In the threshold function (5), the α parameter establishes the initial size, but it is γ which controls the convergence rate. By adaptively adjusting γ, it is possible to control the convergence speed and stabilize the threshold at the "ideal search scale". A simple strategy is used to adjust γ. If an improvement is achieved, i.e. at least one offspring survives to the next generation, it suggests that exploration is paying off and convergence should be delayed – subsequently γ is decreased. Otherwise, if no improvements are achieved, it implies that the current search scale has been sufficiently explored so more local search may now be necessary – subsequently γ is increased.

Fig. 5 compares the standard and adaptive threshold functions. The adaptive threshold is shown for Rastrigin (F15) and the Gallagher (F21) functions. The (initial) threshold parameters are $\alpha = 0.1$ and $\gamma = 3$ and the step size to increase/decrease γ is 0.005. As it can be seen, the threshold value of the adaptive function decreases until it stabilizes at a potentially "ideal search scale". This "ideal" scale is appropriately different for functions with differing characteristics. To guarantee convergence at the end, the threshold size is decreased during the last 20% of the FEs with a constant $\gamma = 3$.

In Table III, the adaptive threshold version (MPS_Ad) is compared to standard MPS, DE, and PSO on functions with $d = 200$. The presented results are the mean errors, the relative performances, and the p-values of a ***t***-test. As it can be seen,

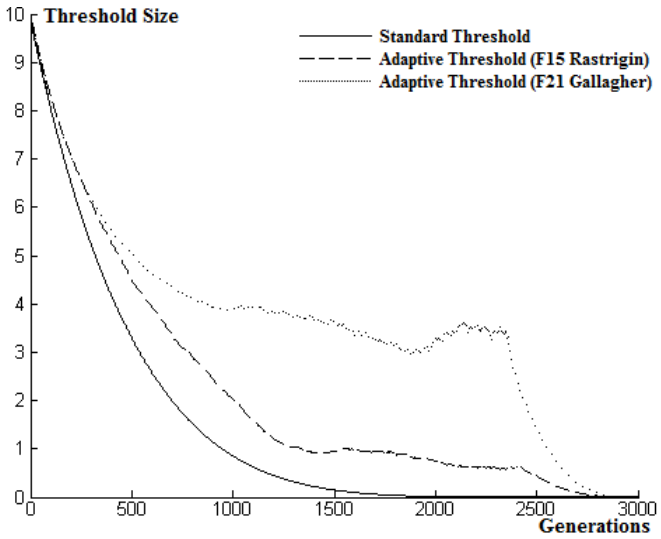

Fig. 5. Standard and adaptive threshold functions.

both MPS versions clearly outperform DE and PSO on most of the functions. Improvements are larger on multi-modal functions with adequate global structure (Set 4), than on functions with weak global structure (Set 5). This result suggests that MPS's search strategy, based on a methodical and full-dimensional search, is capable of detecting and exploiting the regularities underlying the function's structure. In general, the worst comparative results occur for functions 21 and 22 (Gallagher's Gaussian functions [26]). As specified in [21], the key property of these functions is the existence of optima "with position and height being unrelated and randomly chosen". This random feature in these search spaces presents a difficult challenge for adapting the threshold size.

Compared to DE, both MPS versions achieve meaningful and significant improvements (*%-diff* > 10% and $p < 0.05$ for the ***t***-test) on all the multi-modal functions. The adaptive version of MPS outperforms PSO on 9 of the 10 functions. Both MPS versions achieve significant improvements over PSO on 8 of the 10 functions (including all 5 of the functions in Set 4 with adequate global structure). The use of the adaptive threshold improves MPS on 8 of the 10 functions for an overall improvement of 12.3%. Compared to DE and PSO, the adaptive MPS provides an overall improvement of 67.2% and 58.0%, respectively.

VI. Discussion

Line segments formed among the population members are at the core of many population based heuristics such as DE, PSO, and MPS. Line segments are useful to detect and exploit search gradients, especially when differentiation is not possible. However, line segments also require a population large enough to guarantee full coverage of the search space.

With increasing dimensions, larger populations are necessary to maintain good coverage of an exponentially increasing search space. However, if the balance between exploration and exploitation is not properly adjusted, then large populations may not leave enough FEs to achieve convergence (see Fig. 3). This is one of the reasons why a constant population size is a popular guideline for the design of metaheuristics. However, with a fixed population size, coverage becomes sparser as dimensions increase. Further, as population size becomes smaller in comparison to the dimensionality, line segments will get trapped in an n-1 dimensional hyperplane which can become a very small subspace with respect to the whole search space. Without the appropriate search mechanisms, the ability to search beyond the hyperplane will then continuously decrease (see Fig. 4).

Matching the population to the minimum size necessary to fully cover the search space is a key feature of MPS. Taking a step orthogonal to the n-1 dimensional hyperplane explicitly considers the need to continuously explore in all d dimensions of the search space. However, if not enough FEs are left for convergence, a population size which increases with dimensionality may become a liability. The purpose of thresheld convergence is to address this limitation by controlling (through the α and γ parameters) the transition from global to local search. If correctly adjusted, the threshold function can guarantee that enough FEs are left to achieve convergence.

In higher dimensional problems, larger populations

TABLE III
Towards Large Scale Optimization

Set	F	**MPS_Adaptive** mean error	**MPS** mean error	%-diff	p-value	**DE** mean error	%-diff	p-value	**PSO** mean error	%-diff	p-value
4	15	**3.52e+02**	3.91e+02	-9.9%	0.00	2.25e+03	-84.3%	0.00	2.33e+03	-84.8%	0.00
	16	9.61e+00	**9.07e+00**	5.5%	0.02	5.14e+01	-81.3%	0.00	3.56e+01	-73.0%	0.00
	17	**6.85e−01**	7.64e−01	-10.3%	0.00	1.59e+00	-56.9%	0.00	7.66e+00	-91.0%	0.00
	18	**3.47e+00**	3.95e+00	-12.2%	0.00	5.21e+00	-33.4%	0.00	2.87e+01	-87.9%	0.00
	19	**1.45e+00**	2.32e+00	-37.4%	0.01	9.38e+00	-84.5%	0.00	1.28e+01	-88.6%	0.00
Mean set 4				**-12.9%**			**-68.1%**			**-85.1%**	
5	20	**2.20e+00**	2.27e+00	-2.9%	0.01	2.24e+01	-90.1%	0.00	2.40e+00	-8.2%	0.01
	21	**1.77e+00**	2.26e+00	-21.5%	0.04	2.91e+00	-39.1%	0.01	1.92e+00	-7.7%	0.13
	22	4.25e+00	4.07e+00	4.2%	0.14	6.29e+00	-32.4%	0.00	**2.72e+00**	35.9%	0.00
	23	**2.78e−01**	4.43e−01	-37.2%	0.01	3.47e+00	-91.9%	0.00	3.14e+00	-91.1%	0.00
	24	**5.20e+02**	5.25e+02	-1.0%	0.09	2.33e+03	-77.7%	0.00	3.07e+03	-83.0%	0.00
Mean set 5				**-11.7%**			**-66.3%**			**-30.8%**	
Total mean				**-12.3%**			**-67.2%**			**-58.0%**	

Mean error from known optimum and relative improvements of MPS, DE, and PSO versus the Adaptive MPS.

provide information that can be used to develop adaptive threshold functions and improve performance (see Table III). Enhancing MPS through an adaptive threshold function leads to a better balance of global and local search according to the dimensionality and characteristics of the problem. Although the proposed adaptive threshold is an important step toward detecting the "ideal search scale", further improvement may be possible. Since the distance among attraction basins may differ according to the search space region, future work may aim to maintain individual threshold functions for each member (or subset) of the population.

DE and PSO have had many enhancements since their original design, including standard PSO. Since the initial development of MPS is already highly competitive, further enhancements are very promising. In order to increase performance in LSGO, an important research direction will be hybridizing MPS with search techniques with a stronger local-search/exploitation strategy. Once the threshold step achieves the "ideal search scale" and MPS's population finds promising attraction basins, the convergence toward the corresponding (local) optima could be performed through more efficient gradient exploitation techniques. Further improvements may also be achieved through multiple restarts (resizing the minimum and maximum steps) and the combination of MPS with decomposition techniques (such as cooperative coevolution) [9].

VII. Summary

Building up from the original two-dimensional design, this paper extends Minimum Population Search towards large scale global optimization. MPS increases the population to the minimum necessary size in order to guarantee full coverage and continuous exploration in all d dimensions of the search space. With a linearly increasing population, it becomes crucial to correctly balance exploration and exploitation to ensure convergence (to any local optima). MPS adjusts this balance by using thresheld convergence. A new enhancement allows adaptively adjusting the decay rate as the search proceeds so that MPS can stabilize the threshold size at a hopefully "ideal search scale". Altogether, Minimum Population Search is shown to be a simple and effective algorithm for solving complex multi-modal problems. The initial development of MPS is already highly competitive with well-established metaheuristics such as DE and PSO, so future improvements such as hybridization with local search and decomposition techniques are very promising.

References

[1] S. Klein, M. Staring, and J. Pluim, "Evaluation of optimization methods for nonrigid medical image registration using mutual information and b-splines," *IEEE Trans. Image Processing*, vol. 16, pp. 2879–2890, Dec. 2007.

[2] F. Luna, A. J. Nebro, E. Alba, and J. J. Durillo, "Solving large-scale real-world telecommunication problems using a grid-based genetic algorithm," *Engineering Optimization*, vol. 40, pp. 1067–1084, Nov. 2008.

[3] J. F. Chang and P. Shi, "Using investment satisfaction capability index based particle swarm optimization to construct a stock portfolio," *Information Sciences*, vol. 14, pp. 2989–2999, July 2011.

[4] R. E. Bellman, *Dynamic Programming*, Princeton, NJ: Princeton University Press, 1957.

[5] X. Li, K. Tang, M. N. Omidvar, Z. Yang, and K. Qin, "Benchmark Functions for the CEC'2013 Special Session and Competition on Large-Scale Global Optimization," Nature Inspired Computation and Applications Laboratory, USTC, China, Tech. Rep, 2012.

[6] W. Chu, X. Gao, and S. Sorooshian, "A new evolutionary search strategy for global optimization of high-dimensional problems," *Information Sciences*, vol. 181, pp. 4909–4927, Nov. 2011.

[7] A. Bolufé-Röhler and S. Chen, "Minimum Population Search – Lessons from building a heuristic technique with two population members," in *Proc. IEEE CEC*, 2013, pp. 2061–2068.

[8] V. Gardeux, R. Chelouah, P. Siarry, and F. Glover, "EM323 : A Line Search based algorithm for solving high-dimensional continuous non-linear optimization problems," *Soft Computing*, vol. 15, pp 2275–2285, Nov. 2011.

[9] M. Potter and K. De Jong, "Cooperative Coevolution: An Architecture for Evolving Coadapted Subcomponents," *Evolutionary Computation*, vol. 8, pp. 1–29, Mar. 2000.

[10] X. Li and X. Yao, "Cooperatively coevolving particle swarms for large scale optimization," *IEEE Trans. Evolutionary Computation*, vol. 16, pp. 210–224, June 2011.

[11] M. N. Omidvar, X. Li, X. Yao, "Cooperative Co-evolution with Delta Grouping for Large Scale Non-separable Function Optimization," in *Proc. IEEE CEC*, 2010, pp. 1762–1769.

[12] Y. Wang and B. Li, "Two-stage based Ensemble Optimization for Large-Scale Global Optimization," in *Proc. IEEE CEC*, 2010, pp. 4488–4495.

[13] J. Brest, A. Zamuda, B. Boskovic, M. S. Maucec, and V. Zumer, "High-dimensional Real-parameter Optimization Using Self-adaptive Differential Evolution Algorithm with Population Size Reduction," in *Proc. IEEE CEC*, 2008, pp. 2032–2039.

[14] S. Zhao, J. J. Liang, P. N. Suganthan, and M. F. Tasgetiren, "Dynamic Multi-swarm Particle Swarm Optimizer with Local Search for Large Scale Global Optimization," in *Proc. IEEE CEC*, 2008, pp. 3845–3852.

[15] J. C. Lagarias, J. A. Reeds, M. H. Wright, and P. E. Wright, "Convergence properties of the Nelder-Mead simplex method in low dimensions," *SIAM J. Optimization*, vol. 9, pp. 112–147, 1998.

[16] R. Storn and K. Price, "Differential Evolution – A simple and efficient heuristic for global optimization over continuous spaces," *J. Global Optimization*, vol. 11, pp. 341–359, Dec. 1997.

[17] D. Bratton and J. Kennedy, "Defining a standard for particle swarm optimization," in *Proc. IEEE SIS*, 2007, pp. 120–127.

[18] J. Montgomery and S. Chen, "A simple strategy for maintaining diversity and reducing crowding in differential evolution," in *Proc. IEEE CEC*, 2012, pp. 2692–2699.

[19] S. Chen and J. Montgomery, "A simple strategy to maintain diversity and reduce crowding in particle swarm optimization," in *Proc. Australasian AI*, 2011, pp. 281–290.

[20] A. Bolufé-Röhler and S. Chen, "Minimum Population Search – A Scalable Metaheuristic for Multi-Modal Problems", in press.

[21] N. Hansen, S. Finck, R. Ros, and A. Auger, "Real-parameter black-box optimization benchmarking 2009: noiseless functions definitions," INRIA Tech. Rep. RR-6829, 2009.

[22] A. Engelbrecht, "Particle swarm optimization: velocity initialization," in *Proc. IEEE CEC*, 2012, pp. 70–77.

[23] S. Helwig, J. Branke, and S. Mostaghim, "Experimental Analysis of Bound Handling Techniques in Particle Swarm Optimization," *IEEE Trans. Evolutionary Computation*, vol. 17, pp. 259–271, Apr. 2013.

[24] R. Storn, "On the usage of differential evolution for function optimization," in *Proc. IEEE NAFIPS*, 1996, pp. 519–523.

[25] A. Bolufé-Röhler, S. Estévez-Velarde, A. Piad-Morffis, S. Chen and J. Montgomery, "Differential evolution with thresheld convergence," in *Proc. IEEE CEC*, 2013, pp. 40–47.

[26] M. Gallagher and B. Yuan, "A General-Purpose Tunable Landscape Generator," *IEEE Trans. Evolutionary Computation*, vol. 10, pp. 590–603, Oct. 2006.