Optimising real parameters using the information of a mesh of solutions: VMO algorithm

Amilkar Puris and Rafael Bello
Department of Computer Science
University of Las Villas, Cuba
Email: {ayudier, rbellop}@uclv.edu.cu

Daniel Molina
Department of Computer Science and Engineering
University of Cadiz, Spain
Email: daniel.molina@uca.es

Francisco Herrera
Department of Comp. Sci. and A.I.
University of Granada, Spain
Info Contact: http://decsai.ugr.es/ herrera/
Email: herrera@decsai.ugr.es

***Abstract*—Population-based Meta-heuristics are algorithms that can obtain very good results for complex continuous optimisation problems, using the information of a population of solutions. In these algorithms the distribution of solutions is crucial because it has a strong influence of the exploration new regions. In this work, we present a population algorithm, Variable Mesh Optimisation (VMO), in which a set of nodes (potential solutions) is distributed as a mesh. This mesh is initially homogeneously distributed, and then the mesh evolves to a heterogeneous structure resampling the space toward the best neighbours, maintaining at the same time a controlled diversity (avoiding solutions too close to each other). We use a benchmark of multimodal continuous functions to study the influence of the different components of the proposal, and to compare the proposed algorithm with other basic population-based meta-heuristics in the literature. The results show that VMO is a very competitive algorithm.**

***Index Terms*—continuous optimisation, meta-heuristics, population meta-heuristics, variable mesh optimisation**

I. Introduction

Population based meta-heuristics (PMHs) are meta-heuristics that use a solution sets (called population) that evolves during the iterations of the algorithms, using the information of these solutions to make a heuristic sampling of the domain search. Real coded Genetic Algorithms (GAs) [1], Particle Swarm optimisation (PSO) [2], Estimation of Distribution Algorithms (EDAs), Scatter Search (SS) [3], Difference Evolution (DE) [4] are, among others, examples of PMHs.

PMHs introduce different ways of exploring the search space. They present powerful communication or cooperation mechanisms (depending on the context) in order to converge the population toward promising regions of the search domain.

In these mechanisms, the best solutions usually have a strong influence over the remaining ones of the population. For instance, they could have a greater probability of survival into the population, like in genetic algorithms [5]. In other algorithms, remaining solutions are oriented to the best ones directly, as in PSO [2], or more indirectly, as in DE [4], [6].

This type of meta-heuristics, also implements several mechanisms to introduce or maintain diversity into the population. This combination of a convergence mechanism with strategies to introduce the diversity allow PMHs to obtain very good results in continuous optimisation problems.

With these two facts in mind, we present a new PMH called Variable Mesh optimisation, VMO, for real parameter optimisation. In this algorithm, the population is represented by a set of nodes (potential solutions) that are initially distributed as a mesh, using a uniform distribution. Then, VMO evolves this mesh creating more solutions around the most promising regions of the mesh, creating a heterogeneous structure of the mesh. This heterogeneous structure can explore with a better performance, as it is said in [7]. VMO differs from other similar PMHs in the fact that in each iteration it creates new solutions around the current solutions of the mesh, and not only around the best ones, dividing the mesh in more solutions in the most promising regions. Also, to avoid the risk of premature convergence that could be associated with this type of structures, VMO maintain the population diversity by selecting the best representatives of the mesh for the next iteration. The search process developed by VMO can be described by the following two operations:

- Expansion: this mechanism explores around the best solutions found, by creating new nodes between each node of the mesh and its best neighbour, and around the external borders of the mesh.
- Contraction: a clearing process removes all nodes that are too close to others with best fitness. The aim is to maintain the population size and to foment mesh diversity.

We study, based on experimental work, the influence of its different components, showing that the ideas underlying this technique can lead to successful results. Then, we compare the performance of the VMO with other basic PMHs with multimodal functions on continuous domains, showing that VMO is a competitive model.

This paper is organised as follows: In Section II, a detailed description of the VMO is given, emphasizing the expansion and contraction processes of the mesh. In Section III, the experimental framework and the statistical tests used to validate the experimental results are presented. In Section IV, we analyse the behaviour of several VMO components. In Section V, the proposed model is compared with others, to probe if its results improve the obtained by other PMHs of the literature. Finally, in Section VI, we present the main conclusions and

suggest future work.

II. Variable Mesh optimisation

VMO is a PMH in which the population is distributed as a mesh. This mesh is composed of P nodes $(n_1, n_2, \ldots, n_P)$ that represent solutions in the search space. Each node is coded as a vector of M floating point numbers, $n_i = (v_1^i, v_2^i, \ldots, v_M^i) = v_j^i,\ j = 1, \ldots, M$ that represent the solution to the optimisation problem. In the search process developed by VMO, two operations are executed: the expansion and contraction processes. Both processes introduce a suitable balance between exploration and diversity for the VMO algorithm. In following subsections, these operators are described in detail, and in Figure 1 the global algorithm is presented, using the parameters explained in Table I.

Parameter	Description
P	Number of nodes of the population for each iteration
T	Number of new nodes required in the expansion process
k	Number of nodes that define the neighbourhoods of each node of the mesh
C	Algorithm stop condition (maximum number of fitness evaluations)

TABLE I
Parameters of the VMO algorithm

A. Mesh expansion operation

The algorithm develops the expansion process by moving the population through the search space. For this action, new nodes are obtained, using the current population of each iteration, according to the following steps:

Step 1. (*Initial mesh generation*) The initial population for the first algorithm iteration is composed of P nodes that are randomly generated with uniform distribution.

Step 2. (*Nodes generation towards local extremes in neighbourhood*) The first kind of exploration in VMO is carried out in the neighborhood of each node (n_i). The neighborhood of n_i is composed of the k nodes closest (in terms of distance) to it. The best node (fitness) in the neighborhood is selected as the local extreme (n_i^*). Only when n_i^* is better than n_i, a new node is generated between n_i and n_{i^*}. Z new nodes are created, where $Z \leq P - 1$. To detect the neighbourhood of the i-*th* node, we used the euclidean distance (see Equation 1)

$$D_{euclidean}(n_1, n_2) = \sqrt{\sum_{j=1}^{M} (v_j^1 - v_j^2)^2} \tag{1}$$

The new nodes ($n_z = \{v_1^z, v_2^z, \ldots, v_M^z\}$) are calculated using Equation 2.

$$v_j^z = \begin{cases} \overline{m_j}, & \textbf{if } |\overline{m_j} - v_j^{i^*}| > \xi_j \text{ and } \\ & U[0,1] \leq Pr(n_i, n_i^*) \\ v_j^{i^*} + U[-\xi_j, \xi_j], & \textbf{if } |\overline{m_j} - v_j^{i^*}| \leq \xi_j \\ U[v_j^i, \overline{m_j}], & \textbf{othercase} \end{cases} \tag{2}$$

where $\overline{m_j} = average(v_j^i, v_j^{i^*})$, $U[x, y]$ denotes a random value (uniformly) in the interval $[x, y]$.

Pr is the near factor and represents the relation between the fitness of the current node and its local extreme. It takes a value in the range [0, 1], higher when better fitness has n_i. This factor is calculated by Equation 3.

$$Pr(n_i, n_i^*) = \frac{1}{1 + |fitness(n_i) - fitness(n_i^*)|} \tag{3}$$

The variable ξ_j defines the minimum allowed distance for each component, and its value decreases during the running of the algorithm, calculated by Equation 4.

$$\xi_j = \begin{cases} \frac{range(a_j, b_j)}{4} & \text{if } \ c < 0.15\%C \\ \frac{range(a_j, b_j)}{8} & \text{if } \ 0.15\%C \leq c < 0.3\%C \\ \frac{range(a_j, b_j)}{16} & \text{if } \ 0.3\%C \leq c < 0.6\%C \\ \frac{range(a_j, b_j)}{50} & \text{if } \ 0.6\%C \leq c < 0.8\%C \\ \frac{range(a_j, b_j)}{100} & \text{if } \ c \geq 0.8\%C \end{cases} \tag{4}$$

where C and c denote a maximum number of fitness evaluations allowed and the current number of fitness evaluations. In addition, the $range(a_j, b_j)$ denotes the domain amplitude (a_j, b_j) of each component.

The mesh expansion behaves as follows: in the first case the average value between the current node and the local extreme is obtained for the j-th component. In the second case, the local extreme neighbourhood is displaced depending on a distance value for current iteration. And in the last case, a random number is generated between the average value and the current node. Figure 2 shows an example of this step, with $k = 4$.

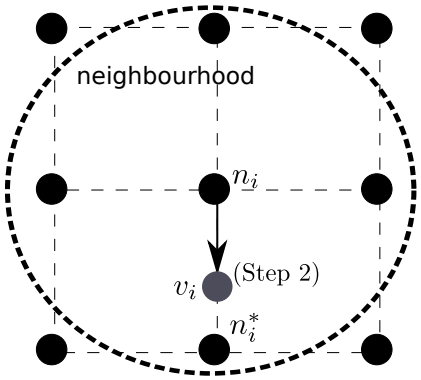

Fig. 2. Example of Step 2

Step 3. (*Nodes generation towards the global extreme*) This step is used to accelerate the algorithm convergence. Thus,

(Define input parameter setting: P, T, k and C, described in Table I).

1) P nodes are randomly generated to compose the initial population and all are evaluated to select the best n_g one.

Expansion

2) For each node of the current population n_i (Z new nodes are created, $Z < P$):
   a) find its closest k nodes using euclidean distance by Equation 1.
   b) select the best node n_i^* (fitness) on neighbours.
   c) if n_i^* is better than n_i, then
      - calculate near factor Pr between n_i and n_i^* by Equation 3.
      - generate a new node by Equation 2 using n_i, n_i^* and Pr.
3) For each node n_i (X new nodes are created, $X = P - 1$):
   a) calculate near factor Pr between n_i and n_g by Equation 3.
   b) create a new node by Equation 5 using n_i, n_g and Pr.
4) If $(Z + X) < T$, select the frontier nodes (external n_s and internal n_u nodes) (Y new nodes are created, $Y = T - (Z + X)$):
   a) calculate w_j displacement vector by Equation 8.
   b) generate $\lfloor Y/2 \rfloor$ new node from n_s by Equation 6.
   c) generate $Y - \lfloor Y/2 \rfloor$ new node from n_u by Equation 7.

Contraction

5) Merge and sort the current population with the created nodes in Steps 2-4 according to their fitness.
6) Apply an adaptive clearing operator, eliminating the near nodes. We obtain B nodes for the next iteration.
7) If $B < P$ then
   - select the B nodes and complete the population with $P - B$ new random nodes.

   Otherwise ($B >= P$) then
   - select the P best nodes (fitness) to compose the population of the next iteration.

8) Go to **Step 2** if the stop condition is not accomplished ($c < C$).

Fig. 1. Steps to apply the VMO algorithm

it explores in the direction of the node which has the best fitness of the current population, called the global extreme of the mesh (n_g). X new nodes $n_x = \{v_1^x, x_2^x, \ldots, x_M^x\}$ are generated ($X = P-1$), one for each n_i, towards n_g, following the Equation 5.

$$v_j^x = \begin{cases} average(v_j^i, v_j^g), & \text{if} \quad U[0,1] \leq Pr(n_i, n_g) \\ U[average(v_j^i, v_j^g), v_j^g], & \text{otherwise} \end{cases} \tag{5}$$

If there is a great difference (in fitness) between n_i and n_g, then there will be a high probability for the j-th component to take closer values to v_j^g. In the other case, the average value is obtained. Figure 3 shows a example of this step.

Step 4. (*Nodes generation starting from the frontier nodes of mesh*) In this step, new nodes are created from the frontier nodes in the current population, to complete the expansion process. This step is only run if the number of created nodes in the previous steps ($Z + X$) is lower than the expected one, T. $Y = T - (Z + X)$ new nodes are created in this step. If $Y > P$, only P new nodes are created, one for each mesh node.

In this step, we consider frontier nodes (n_t). The frontier is composed of the nodes that are nearest (*known as interior frontier or internal nodes*, n_u) and furthest (*known as exterior frontier or external nodes*, n_s) from the point that represents the search space center. To detect these nodes the euclidean distance is used. Starting from these sets, new nodes are created (one for each frontier node), in function of the frontier type.

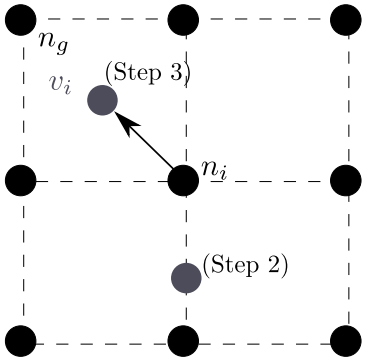

Fig. 3. Example of Step 3

- $\lfloor Y/2 \rfloor$ are created from n_s nodes, using the Equation 6.

$$v_j^h = \begin{cases} v_j^s + w_j, & \text{if} \quad v_j^s > 0 \\ v_j^s - w_j, & \text{if} \quad v_j^s < 0 \end{cases} \tag{6}$$

- $Y - \lfloor Y/2 \rfloor$ nodes are created from n_u nodes, following the Equation 7.

$$v_j^h = \begin{cases} |v_j^u + w_j|, & \text{if} \quad v_j^u > 0 \\ |v_j^u - w_j|, & \text{if} \quad v_j^u \leq 0 \end{cases} \tag{7}$$

where w_j represents a displacement for each component j and is calculated in a decreasing way in function to the running of the algorithm, according to Equation 8:

$$w_j = (w_j^0 - w_j^1) \cdot \frac{C - c}{C} + w_j^1 \tag{8}$$

The variable w_j take values between w_j^0 w_j^0 represents the initial displacement and w_j^1 its final value. In addition, the values C and c are used (see the description of Equation 4). In this work we use $w_j^0 = range(a_j, b_j)/10$ and $w_j^1 = range(a_j, b_j)$ $/100$. Figure 4 shows an example of this step, with $Y = 4$.

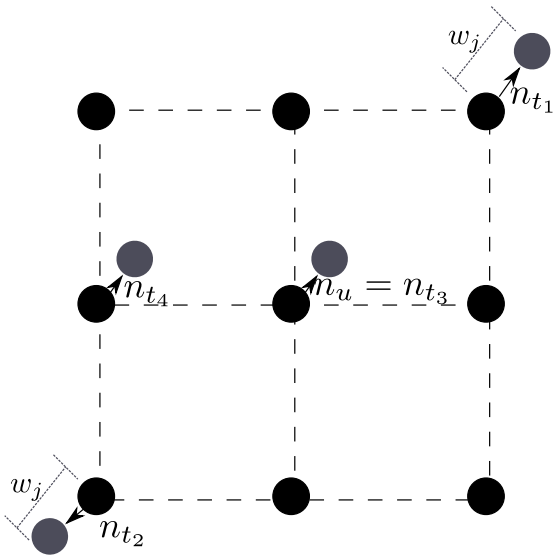

Fig. 4. Example of Step 4

B. Mesh contraction process

The contraction operation selects the nodes of the mesh that will be used as the population for the next algorithm iteration. Nodes with the best fitness are selected from among the current mesh and the new nodes created for the expansion process, applying an elitist criterion.

Before the selection, VMO applies an adaptive clearing operator to increase diversity in the mesh in order to keep a minimum distance between the mesh nodes.

In the following the contraction process is described:

Step 5. All mesh nodes are ordered depending on their fitness (Ascending).

Step 6. The difference between each node and their successors is calculated for each dimension. Successor nodes with any difference smaller than its corresponding ξ_j value are removed. The ξ_j value is calculated by Equation 4. Finally we are left with B nodes.

Step 7. The nodes with best fitness are selected as the population for the next iteration. If $B < P$ then the mesh is completed with new random nodes.

This mechanism considers two important elements: the node qualities and their places in the solution space. The nodes with better fitness have a higher probability of taking part in the next population. Adaptive clearing allows the method to carry out more general explorations and eventually to reduce its frequency to focus on a smaller search space area. This element increases the method's exploitation level and makes it stronger.

III. Experimental Framework

The test suite that we have used for the experiments consists of 20 benchmark multimodal functions chosen from the set designed for the special session on real parameter optimisation organised in the 2005 IEEE congress on evolutionary computation (CEC2005) [8]. The unimodal functions are not used for this study because these are simpler than the multimodal functions. In [8] they can be found the complete description of the multimodal functions and their source codes.

In order to be able to compare our results with other algorithms involved in the competition, we followed the requirements described in [8]:

- Each algorithm is run 25 times for each test function, and the average of error of the best individual of the population is computed. The *function error* value for a solution x is defined as $(f(x) - f(x^*))$, where x^* is the global optimum of the function.
- The study has been made with dimensions $D = 10$, $D = 30$, and $D = 50$.
- The maximum number of fitness evaluations for each run (parameter C) is $10,000 \cdot D$, where D is the dimension of the problem.
- Each run stops either when the error obtained is less than 10^{-8}, or when the maximal number of evaluations is achieved.

We have used for the comparisons the non-parametric tests [9], because it was recommended for the used test suite [10]. In particular, we use the statistical tests recommended in [11]:

- Application of the Wilcoxon matched-pairs signed-ranks test, to compare directly the differences between two algorithms.
- Application of Iman and Davenport's test and Holm's method as post-hoc procedures, to compare three or more algorithms. The first test is used to see whether there are significant statistical differences among the algorithms to compare. If differences are detected, then Holm's test is employed to compare the best ranking algorithm (control algorithm) with the remaining ones.

Any reader interested in this topic can find additional information on the Web site http://sci2s.ugr.es/sicidm/.

IV. Study of the VMO's parameter

In this section, we study the VMO's behaviour for some internal elements: size of the initial population (Section IV-A), adaptive clearing operator (Section IV-B) and generation by means of the frontier's nodes (Section IV-C).

A. Size of mesh

The definition of the size of the population is very important in current PMHs in order to obtain a high performance from these methods. In the specific case of VMO, is specially important, because all the population nodes are involved in the explorations.

In this section, we study the results with different sizes of mesh: P =(8, 12, 24, 50, and 100). In all cases, the total expansion size used for these studies is $T = 1.5 \cdot P$. The neighbourhood size is fixed to $k = 3$.

To apply the non-parametric test to this case study, we present in Table II the average ranking of the VMO algorithm for different mesh sizes for all dimensions (the VMO(P) refers to the VMO algorithm with P population size, the best rank is presented in bold typeface). The results of Iman-Davenport's test show that the mesh size could produce significant differences; due to this fact the hypothesis of equality has been rejected for each dimension, because the statistical test value is greater than the critical value (see Table III).

TABLE II
RANKING OF THE VMO ALGORITHM FOR DIFFERENT MESH SIZES FOR EACH DIMENSION.

Algorithm	*Rank (D=10)*	*Rank (D=30)*	*Rank (D=50)*
VMO(8)	4.4750	3.449	2.875
VMO(12)	3.650	**1.400**	**1.400**
VMO(24)	2.900	2.600	2.775
VMO(50)	**1.651**	3.950	3.250
VMO(100)	2.325	3.600	4.700

TABLE III
RESULTS OF IMAN-DAVENPORT'S TEST FOR DIFFERENT MESH SIZES FOR EACH DIMENSION.

Dimension	*Test value*	*Critical value*	*Hypothesis*
D=10	18.154	2.4920	**Rejected**
D=30	13.674	2.4920	**Rejected**
D=50	23.974	2.4920	**Rejected**

Holm's test is applied to compare the configuration with the best rank in each dimension (VMO(50) for $D = 10$ and VMO(12) for $D = 30$ and $D = 50$), with each of the four remaining ones. Table IV contains all the computations associated with Holm's procedure.

Table IV shows that results obtained with a mesh size of 50 nodes are significantly superior to three of the compared configurations (VMO(8), VMO(12) and VMO(24)) for $D = 10$. Only in comparison with VMO(100) are the differences insignificant. For $D = 30$ and $D = 50$ the VMO algorithm with a population size of 12 nodes obtained significantly superior results to the other mesh configurations.

In these experiments we can conclude that the mesh size has a influence over the results, and its bet value depends on the dimension of the problem. For problems with a small dimension a population size of between 50 and 100 nodes obtains good results, and for high dimensions the best results are obtained with a smaller mesh.

In the following experiments, we use a popsize $P = 12$ for dimension 10, and $P = 50$ for dimensions 30 and 50.

TABLE IV
RESULTS OF HOLM'S TEST FOR EACH DIMENSION WITH P-VALUE=0.05

i	*Algorithm*	$z = \frac{R_0 - R_i}{SE}$	*p-value*	α/i	*Hypothesis*
	D=10, VMO(50) as reference algorithm				
4	VMO(8)	5.649	1.60E-08	0.0125	**Rejected**
3	VMO(12)	3.999	6.33E-05	0.0166	**Rejected**
2	VMO(24)	2.499	0.012	0.0250	**Rejected**
1	VMO(100)	1.349	0.177	0.0500	**Accepted**
	D=30, VMO(12) as reference algorithm				
4	VMO(50)	5.100	3.39E-7	0.0125	**Rejected**
3	VMO(100)	4.400	1.08E-5	0.0166	**Rejected**
2	VMO(8)	4.099	4.13E-5	0.0250	**Rejected**
1	VMO(24)	2.400	0.016	0.0500	**Rejected**
	D=50, VMO(12) as reference algorithm				
4	VMO(100)	6.600	4.11E-11	0.0125	**Rejected**
3	VMO(50)	3.699	2.15E-4	0.01666	**Rejected**
2	VMO(8)	2.949	0.003	0.0250	**Rejected**
1	VMO(24)	2.750	0.005	0.0500	**Rejected**

B. Adaptive clearing operator

Here we are going to study the effect of applying an adaptive clearing operator to the VMO algorithm. For this study, we compare the results among other clearing operators and our adaptive proposal. VMO-NC and VMO-AC represent the results of the algorithm without the clearing operator, and using the adaptive clearing, respectively. VMO-C1, VMO-C2, VMO-C3, VMO-C4 y VMO-C5 show the VMO solutions with the clearing operator for different constant values of the distance ($\xi_j = \frac{range(a_j,b_j)}{4}$, $\frac{range(a_j,b_j)}{8}$, $\frac{range(a_j,b_j)}{16}$, $\frac{range(a_j,b_j)}{50}$ and $\frac{range(a_j,b_j)}{100}$, respectively), depending on the domain amplitude in each test function (see Equation 5).

TABLE V
RESULTS OF THE IMAN-DAVENPORT'S TEST FOR DIFFERENT ALTERNATIVES OF CLEARING OPERATOR FOR EACH DIMENSION.

Dimension	*Test value*	*Critical value*	*Hypothesis*
D=10	26.315	2.1791	**Rejected**
D=30	64.320	2.1791	**Rejected**
D=50	14.641	2.1791	**Rejected**

First, we apply Iman-Davenport's test to each dimension, Table V shows the results, detecting statistically significant differences for each dimension. Thus, we applied Holm's multiple comparisons test, to find out which algorithm is statistically better than the others.

Holm's test compares the algorithm with the best rank for each dimension VMO-AC, with each one of the other configurations, in pairs. Table VI shows the results of Holm's test with the significance value 0.05. We can see that there are significant differences in the majority of cases, with the exception of VMO-C3 and VMO-C2, where there are no significant differences in dimension 10.

We have also applied Wilcoxon's test to determine which of them presents the best behaviour. Table VII shows the results between VMO-AC and VMO-C3, and VMO-AC and VMO-C2 for $D = 10$, showing that VMO-AC is statistically better. In resume, not only the clearing method improves the results, but also the proposed adaptive clearing method statistically improves results obtained with a fixed distance value.

TABLE VI
RESULTS OF THE HOLM'S TEST FOR EACH DIMENSION, WITH P-VALUE=0.05

i	*Algorithm*	$z = \frac{R_0 - R_i}{SE}$	*p-value*	α/i	*Hypothesis*
	D=10, VMO-AC as reference algorithm				
6	VMO-NC	6.733	1.65E-11	0.0080	**Rejected**
5	VMO-C5	5.379	7.46E-8	0.0100	**Rejected**
4	VMO-C4	3.989	6.63E-5	0.0125	**Rejected**
3	VMO-C1	3.440	5.86E-4	0.0166	**Rejected**
2	VMO-C3	1.427	0.153	0.0250	**Accepted**
1	VMO-C2	1.061	0.288	0.0500	**Accepted**
	D=30, VMO-AC as reference algorithm				
6	VMO-NC	8.087	6.07E-16	0.0080	**Rejected**
5	VMO-C5	6.587	4.48E-11	0.0100	**Rejected**
4	VMO-C4	5.635	1.74E-8	0.0125	**Rejected**
3	VMO-C1	4.062	4.86E-5	0.0170	**Rejected**
2	VMO-C3	2.927	0.003	0.0250	**Rejected**
1	VMO-C2	2.159	0.031	0.0500	**Rejected**
	D=50, VMO-AC as reference algorithm				
6	VMO-NC	6.661	2.73E-11	0.0080	**Rejected**
5	VMO-C4	4.977	6.45E-7	0.0100	**Rejected**
4	VMO-C5	4.794	1.63E-6	0.0125	**Rejected**
3	VMO-C3	4.391	1.13E-5	0.0167	**Rejected**
2	VMO-C1	3.989	6.65E-5	0.0250	**Rejected**
1	VMO-C2	2.855	0.004	0.0500	**Rejected**

TABLE VII
RESULTS OF WILCOXON'S TEST (SIGNIFICANCE VALUE 0.05)

VMO-AC vs	R^+	R^-	*p-value*	*Hypothesis*
VMO-C3	156.50	**53.50**	0.038	**Rejected**
VMO-C2	167.50	**42.50**	0.013	**Rejected**

C. Generation from the frontiers nodes

This generation process explores the domain space around the frontiers of the population. In this section, we check if this exploration of frontiers improves the search. Then, we compare the experimental results comparins the VMO when the frontier's operator is not used, VMO-NF, against the VMO using the frontier's operator, VMO-F. For this study, we use Wilcoxon's test to compare both algorithms for each dimension.

TABLE VIII
RESULTS OF WILCOXON'S TEST (SIGNIFICANCE VALUE 0.05)

VMO-F vs VMO-NF	R^+	R^-	*p-value*	*Hypothesis*
D=10	190.50	**19.50**	0.001	**Rejected**
D=30	180.50	**29.50**	0.002	**Rejected**
D=50	180.50	**29.50**	0.002	**Rejected**

It can clearly be seen (see Table VIII) that VMO-F obtains better results than VMO-NF in all dimensions (R^+ values are higher than the R^- ones). In addition, the statistical test indicates that these improvements are statistically significant.

V. COMPARATIVE STUDY WITH OTHERS ALGORITHMS

In this section, we present a comparative study between VMO and other PMHs that have demonstrable a high level of performance. These PMHs are representative evolutionary algorithms. We compare with the following algorithms.

- Steady-State Genetic Algorithm (SSGA) [12], with $BLX - \alpha$ operator ($\alpha = 0.5$), selection of parents Negative Assortative Mating [13] with $Nnam = 3$, and a replacement strategy of replacing the worst (RW). The population size was 60 solutions.
- Linearly Decreasing Inertia Weight in Particle Swarm optimisation (LDWPSO) [14]. The algorithm PSO proposed by Shi and Ebenhart is taken into account, applying the authors' configuration: the inertia varies from the maximum value ($w_{max} = 0.9$) to the minimum value ($w_{min} = 0.4$); and the parameters c1 y c2 are equal to 2.8 and 1.3, respectively. The parameters used were defined by the authors themselves, you can see the reference to obtain more information.
- Opposite Differential Evolution (ODE) [15]. This algorithm is a DE that enforces diversity in the search, considering in the search process the opposite of each new solution created. The parameters used were defined by the authors themselves, you can see the reference to obtain more information.

In this analysis we compare VMO with other algorithms presented in this section, using Wilcoxon's test. Tables IX, X and XI show the results for dimension 10, 30, and 50, respectively.

TABLE IX
RESULTS OF WILCOXON'S TEST FOR $D = 10$

VMO vs	R^+	R^-	*p-Value*	*Hypothesis*
SSGA	210.0	**0.0**	0.000	**Rejected**
ODE	**89.0**	121.0	0.550	**Accepted**
LDWPSO	210.0	**0.0**	0.000	**Rejected**

TABLE X
RESULTS OF WILCOXON'S TEST FOR $D = 30$

VMO vs	R^+	R^-	*p-Value*	*Hypothesis*
SSGA	210.0	**0.0**	0.000	**Rejected**
ODE	156.5	**53.5**	0.040	**Rejected**
LDWPSO	210.0	**0.0**	0.000	**Rejected**

TABLE XI
RESULTS OF WILCOXON'S TEST FOR $D = 50$

VMO vs	R^+	R^-	*p-Value*	*Hypothesis*
SSGA	210	**0**	0.000	**Rejected**
ODE	205	**5**	0.000	**Rejected**
LDWPSO	167	**43**	0.044	**Rejected**

According to the results obtained in the tables it can be observed that:

- VMO is significantly better than the SSGA and LDWPSO for $D = 10$. VMO is only worse in absolute terms in relation to ODE ((R^+ values are lower than the R^- ones)), but not in a significant way.
- VMO is statistically better than all considered PMHs into consideration for $D = 30$ and $D = 50$.

VI. Conclusions

A PMH denominated Variable Mesh optimisation (VMO) was introduced in this paper. It includes new ways of exploring the search space:

- It create solutions towards the neighbour with best fitness for each mesh node, towards the node with the best fitness of the mesh, and the frontiers of the mesh. For these exploration methods, the algorithm obtains a suitable balance between the exploitation and the exploration.
- It combines an elitist replacement criterion for population, taking into account the quality and the distance between the nodes by means of a clearing operator. This mechanism introduces diversity in the population that facilitates a larger exploration of the solution space. In addition, the population contains the best representative of each zone of the explored domain search.
- The clearing operator functions in an adaptive manner because it decreases the allowed distance between the nodes as the algorithm is conducted, depending on the extent of each interval. This operator causes the method to start with a high exploration level that decreases as the allowed distance during the running of the algorithm.

It has been shown that the proposed VMO algorithm presents a good scalability level presenting good results with dimensions 30 and 50.

The promising research line initiated with the present optimisation framework based on VMO is worthy of further study. We will extend our investigation to test its behaviour with higher dimensions. Furthermore, we will study the clearing operator to regulate the diversity levels that it introduces in the mesh and to see its influence on the behavior of the VMO algorithm.

References

[1] F. Herrera, M. Lozano, and J. Verdegay, "Tackling realcoded genetic algorithms: Operators and tools for the behavioral analysis," *Artificial Intelligence Reviews*, vol. 12, no. 4, pp. 265–319, 1998.

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

[3] M. Laguna and R. Martí, *Scatter Search. Methodology and Implementation in C*. Kluwer Academic Publishers, 2003.

[4] R. Storn and K. Price, "Differential Evolution: A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces," *Journal of Global Optimization*, vol. 11, pp. 341–359, 1997.

[5] L. Davis, in *Handbook of Genetic Algorithms*, N. Y. Van Nostrand Reinhold, Ed., 1991.

[6] J. Brest, B. Boskovic, S. Greiner, V. Zumer, and M. S. Maucec, "Performance comparison of self-adaptive and adaptive differential evolution algorithms," *Soft Computing*, vol. 11, no. 7, pp. 617–629, 2007.

[7] J. Kennedy and R. Mendes, "Population structure and particle swarm performance," in *Proceeding of the 2002 Congress on Evolutionary Computation*, 2002, pp. 1671–1676.

[8] P. Suganthan, N. Hansen, J. Liang, K. Deb, Y.-P. Chen, A. Auger, and S. Tiwari, "Problem Definitions and Evaluation Criteria for the CEC 2005 Special Session on Real-Parameter Optimization," Nanyang Technological University, http://www.ntu.edu.sg/home/EPNSugan/, Tech. Rep., May 2005.

[9] D. J. Sheskin, *Handbook of parametric and nonparametric statistical procedures*. Chapman and Hall/CRC, 2007.

[10] S. García, D. Molina, M. Lozano, and F. Herrera, "A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the CEC'2005 Special Session on Real Parameter Optimization," *J. Heuristics*, vol. 15, no. 6, pp. 617–644, 2009.

[11] J. Derrac, S. García, D. Molina, and F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms." *Swarm and Evolutionary Computation*, vol. 1, pp. 3–8, 2011.

[12] G. Syswerda, "Uniform Crossover in Genetic Algorithms," in *Proc. Thrid International Conference on Genetic Algorithms*, J. Schaffer, Ed. Morgan Kaufmann, San Mateo, 1989, pp. 2–9.

[13] C. Fernandes and A. Rosa, "A Study of non-Random Matching and Varying Population Size in Genetic Algorithm using a Royal Road Function." in *Proc. IEEE Congress on Evolutionary Computation*, no. 1. IEEE Press, Piscataway, New York, 2001, pp. 60–66.

[14] Y. Shi and C. Eberhart, "A Modified Particle Swarm Optimizer," in *Proc. IEEE International Conference on Evolutionary Computation*, 1998, pp. 69–73.

[15] S. Rahnamayan, H. Tizhoosh, and M. Salama, "Solving large scale optimization problems by Opposition-Based Differential Evolution," *IEEE Transactions on Computation*, vol. 7, no. 10, pp. 1792–1804, 2008.