# Analyzing the Impact of the Adaptive Clearing Mechanism on Algorithm Accuracy in Variable Mesh Optimization

Frank Vanhoenshoven[1], Gonzalo Nápoles[1], Mathijs Creemers[1], Maikel Leon Espinosa[2], and Koen Vanhoof[2]

[1]Faculty of Business Economics, Hasselt University, Campus Diepenbeek, Agoralaan Building D, BE3590 Diepenbeek, Belgium
[2]Department of Business Technology, University of Miami, Coral Gables, FL 33124, USA

2016

**Abstract**

The area of population-based meta-heuristics has been researched extensively in recent years. The focus of this research has been on finding improvements and variations to existing algorithms while the inner details, that are treated as a black box, remain poorly understood. The purpose of this paper is to uncover the detailed behavior of Variable Mesh Optimization (VMO), a population-based meta-heuristic, and describe the patterns that drive the algorithm in finding new optima. Our results suggest that, in VMO, the improvement of the best solution is strongly correlated with its adaptive clearing mechanism. It is observed that each relaxation of the threshold that is used by the mechanism, is likely to increase the accuracy of the final solution. These findings suggest that future research, aiming to improve algorithm accuracy, could focus on improving the adaptive clearing mechanism in order to increase the likelihood of creating superior algorithms.

## 1 Introduction

Recent years have shown extensive research activity in the area of population-based meta-heuristics (pmh). Due to this extensive research, numerous authors have described the fundamentals behind the heuristics [**?**], [**?**], [**?**], [**?**]. The inner workings operate based on the idea that a group of relatively simple individuals are nonetheless able to display complex behavior. Inspiration for these algorithms is drawn from nature and examples are abundant in the behavior of schools of fish, flocks of birds, ant colonies...

This behavior is translated into a population-based meta-heuristic by representing an optimization problem in an n-dimensional solution space and populate it with individuals that each represent a possible solution. The individuals will use the mechanism of the system that has been used as inspiration, to move through the solution space in search of an optimal location.

Amongst others, a comprehensive overview of these heuristics has been published by Yang [**?**]. Judging by the amount of research papers published on the subject, Particle swarm optimization, introduced by Kennedy et al. [**?**] can be considered a popular population-based meta-heuristic. In PSO, particles move in a search space with a certain velocity and try to find an optimal position. In each iteration, this velocity is updated by using information from their surroundings and a memory of personal best positions. In Ant colony optimization [**?**], behavior of the system has been inspired on the nature of an ant colony and the characteristic of ants to drop and use pheromone trails to help with their navigation. Inspiration for the Bee algorithm [**?**] was drawn from the behavior of a bee hive. Other examples can be found in Cuckoo qearch and the Firefly algorithm, which is a more generic form of PSO [**?**].

Variable mesh optimization (VMO), presented by Puris et al. [**?**], [**?**], defines its individuals as nodes that are organized in a mesh. The nodes will generate offspring nodes in promising regions of the search space, aiming to detect the optimal solution. Typical to pmh, the algorithm assumes that those promising regions are located nearby the nodes with the highest fitness values. Therefore, *local best* and *global best* nodes will be used as reference nodes and node generation will be concentrated around them. To promote exploration and avoid early convergence, the algorithm uses what is called the *adaptive clearing mechanism*. *Adaptive clearing* is designed to avoid, especially in the early phases of the algorithm, too dense concentrations of nodes in a single area and imposes a minimal distance between nodes by removing all individuals whose proximity to a better node falls within a certain threshold value. In short, for every given area in the search space, the algorithm will only retain the node with highest fitness value. The shift from exploration towards exploitation is achieved by decreasing that minimal allowed distance at discrete steps throughout the life cycle of the algorithm, allowing nodes to exist closer to each other, and explore the same area of the search space. More details on VMO will be provided in section 2.

Many improvements have been proposed on the population-based meta-heuristics and a lot of the research in this area is focusing on the topic of algorithm improvements. For VMO, effort has been put in modifying the algorithm to deal with multimodal problems [**?**], [**?**], [**?**], [**?**].

Although these alterations of an original implementation often prove to be successful in solving a variety of problem sets, it is still not well understood why some of these modifications generate positive results whereas others do not. A notable exception has been provided by Clerc and Eberhart [**?**], who described the internal behavior of PSO by tracking and examining the movements of individual particles.

This paper attempts to perform a similar research on the VMO algorithm and

gain insights into the inner workings by examining the behavior of the individuals rather than the group. The goal is to identify the internal drivers of algorithm accuracy, hereby highlighting the areas in which future improvements have a higher likelihood of being successful. As VMO is one of the younger pmh's, less modifications already exist, which increases the added value of a better understanding of hidden dynamics.

A priori, we have concerns about the threshold used by *adaptive clearing*. The mechanism will remove nodes in order to promote exploitation, but is therefor likely to remove nodes that could have helped the algorithm in finding a better solution as well. The original implementation [**?**] defines seemingly random moments in which to decrease the adaptive clearing threshold, while also decreasing it by a seemingly random factor. The main focus of this paper will thus concentrate on the impact of the current adaptive clearing mechanism on algorithm accuracy.

The remainder of this paper is organized as follows: Section 2 provides more details about the VMO algorithm that has been devised by Puris et al. [**?**]. The research questions in Section 3 are solved using the methodology that is briefly described in Section 4. Results can be found in Section 5, with the discussion in Section 6. Finally, conclusions and further research scopes will be summarized in Section 7.

# 2 Variable Mesh Optimization

The pseudo-code of the VMO procedure, described in [**?**], can be found in Algorithm 1.

Starting from a random group of individuals, nodes, in a problem set, the VMO algorithm will iterate between expansion and contraction phases until a maximum number of iterations is reached.

In the expansion phase, explained in subsection 2.1, each of the existing nodes will generate new nodes in more promising regions of the problem space. These new nodes will roughly appear halfway between the parent node and a target node. The target nodes represent the promising regions of the search space and are selected based on the best fitness values among the neighboring nodes (local search) and the entire group (global search). To increase the search space, some nodes will generate offspring away from the center of the group, called frontier search.

During the contraction phase, see subsection 2.2, only the most promising nodes are retained for the next iteration. The algorithm aims to favor exploration in the early phases while shifting focus to exploitation towards the end. The *adaptive clearing mechanism* is designed to do exactly this by removing all nodes that fall within a minimum allowed distance of a better node. At discrete points throughout a run, more specifically after 15, 30, 60 and 80% of the maximum allowed number of function evaluations, the minimal allowed distances are decreased and the algorithm is allowed to populate promising areas of the search space more densely. After *adaptive clearing*, the $N$ best performing individuals

**Algorithm 1** Variable Mesh Optimization

Determine Parameters
$N$ = Number of nodes in Mesh
$C$ = Maximum number of function evaluations
$k$ = Number of neighboring nodes used in local search
Generate $N$ nodes at random
Calculate all fitness values
Determine global best $n_g$
**while** Maximum problem evaluations not reached **do**
    Update adaptive clearing threshold $\varsigma$ if needed
    **procedure** LOCAL SEARCH
        **for all** $n \in N$ **do**
            Determine $k$ nearest nodes
            Determine local best $n_l$
            Generate new node between $n$ and $n_l$
        **end for**
    **end procedure**
    **procedure** GLOBAL SEARCH
        **for all** $n \in N$ **do**
            Generate new node between $n$ and $n_g$
        **end for**
    **end procedure**
    **procedure** FRONTIER SEARCH
        Select $T$ nodes from frontier of mesh
        **for all** $n_i \in T$ **do**
            Generate new node outwards
        **end for**
        Select $Z$ nodes from center of mesh
        **for all** $n_i \in Z$ **do**
            Generate new node outwards
        **end for**
    **end procedure**
    Sort all nodes by fitness value
    **procedure** ADAPTIVE CLEARING
        **for all** $n_i \in N$ **do**
            **if** $distance(n_i, n_{i-1}) \leq \varsigma$ **then**
                Remove $n_i$ from $N$
            **end if**
        **end for**
    **end procedure**
    Retain only the $N$ best nodes
    Determine global best $n_g$
**end while**

are selected for the next iteration, while the remaining nodes are discarded.

As explained in subsection 2.3, the algorithm will stop after a defined number of function evaluations has been performed.

## 2.1 Expansion Phase

In the expansion phase, the algorithm will generate new nodes in promising areas of the search space. Creation of these new nodes is handled by three different mechanisms.

First of all, the algorithm will calculate the fitness values for all nodes. Each node will then identify the local best solution by comparing the fitness values of their $k$ nearest neighbors. Once the local best is identified, each other node will generate a new node towards the local best, using Eq. 1 and 2. The value of a new node $x$ in dimension $j$, $v_j^x$, is equal to the average position $m_j$ between its parent node, $v_j^i$ and the parent's best neighbor $v_j^{i^*}$, if that average $m_j$ is further than a threshold $\varsigma_j$ (see Eq. 3) from the local best and if the fitness values of parent node and local best are relatively similar to each other (see Eq. 2). In case the average position $m_j$ would fall within the minimum allowe distance $\varsigma_j$ of the local best, the new coordinate will be randomly located anywhere near the local best. In other cases, the new node will be located somewhere between the average position and the parent node.

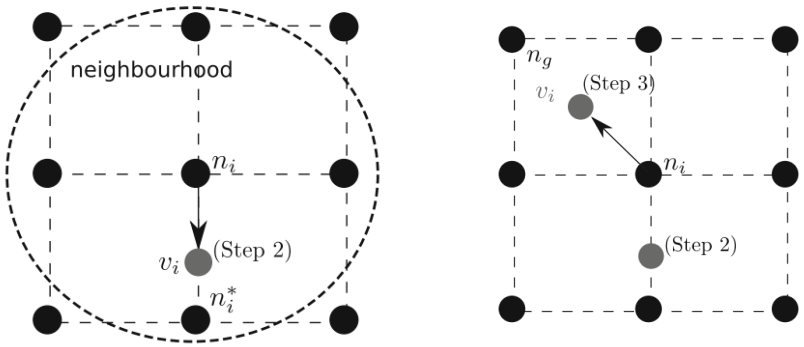
This mechanism is called *local search* and is depicted in Fig. 1a.
Note that the nearness $Pr$ is a a measure of similarity between fitness values of two nodes. The threshold $\varsigma_j$ evolves during the course of the algorithm and is a function of the lowest possible value for dimension j, $a_j$, the highest possible value; $b_j$, as well as the current ($c$) and maximum ($C$) number of fitness evaluations.

$$v_j^x = \begin{cases} \bar{m_j}, & \text{if } |\bar{m_j} - v_j^{i^*}| > \varsigma_j. \\ & \text{AND } U[0,1] \leq Pr(n_i, n_i^*). \\ v_j^{i^*} + U[-\varsigma_j, \varsigma_j], & \text{if } |\bar{m_j} - v_j^{i^*}| \leq \varsigma_j. \\ U[v_j^i, \bar{m_j}], & \text{otherwise.} \end{cases} \tag{1}$$

$$Pr(n_i, n_i^*) = \frac{1}{1 + |\text{fitness}(n_i) - \text{fitness}(n_i^*)|} \tag{2}$$

$$\xi_j = \begin{cases} \frac{range(a_j, b_j)}{4}, & \text{if } c < 0.15C \\ \frac{range(a_j, b_j)}{8}, & \text{if } 0.15C \leq c < 0.30C \\ \frac{range(a_j, b_j)}{16}, & \text{if } 0.30C \leq c < 0.60C \\ \frac{range(a_j, b_j)}{50}, & \text{if } 0.60C \leq c < 0.80C \\ \frac{range(a_j, b_j)}{100}, & \text{if } c \geq 0.8C \end{cases} \tag{3}$$

In *global search* (Fig. 1b), the node with highest fitness is elected as the global best solution. Each other node will then generate a new node, generally in between themselves and the global best, as shown in Eq. 4, where $v_j^i$ represents




(a) Local Search [?]

(b) Global Search [?]

the parent node and $v_j^g$ the global best. Here, the new node will be placed at the exact average location for nodes with a high nearness, $Pr$, and somewhere between the average and the global best for other nodes.

$$v_j^x = \begin{cases} average(v_j^i, v_j^g), & \text{if } U[0,1] \leq Pr(n_i, n_g) \\ U[average(v_j^i, v_j^g), v_j^g], & \text{otherwise} \end{cases} \tag{4}$$

The last mechanism is called *frontier search* and is intended to retain diversity in the mesh. It also detects optimal solutions that are laying outside the ranges of the original search space. In frontier search, a number of nodes will generate extra nodes from the boundaries of the mesh, thus increasing the range of the search area. The mechanism follows Eq. 6 and 7 for nodes on the boundaries of the search space and Eq. 5 and 7 for inner nodes and is depicted in Fig. 2a. As parent nodes, the algorithm will use those that are nearest, $v^g$, and furthest, $v^u$ from the center of the mesh.

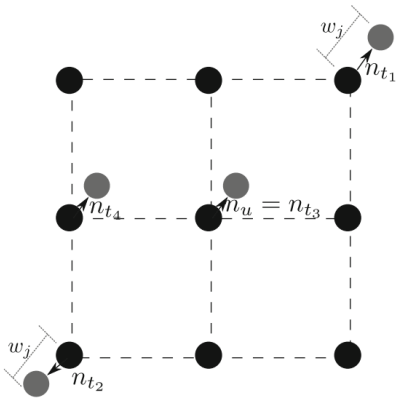
$$v_j^h = \begin{cases} v_j^g + w_j, & \text{if } v_j^g > 0 \\ v_j^g - w_j, & \text{if } v_j^g \leq 0 \end{cases} \tag{5}$$

$$v_j^h = \begin{cases} |v_j^u + w_j|, & \text{if } v_j^u > 0 \\ |v_j^u - w_j|, & \text{if } v_j^u \leq 0 \end{cases} \tag{6}$$

$$w_j = (\frac{range(a_j, b_j)}{10} - \frac{range(a_j, b_j)}{100})\frac{C-c}{C} + \frac{range(a_j, b_j)}{100}) \tag{7}$$

## 2.2 Contraction Phase

During the contraction phase, the algorithm eliminates the nodes that it considers least useful and will continue the next iteration with only the most promising ones. Selection of these most promising nodes is done via two different mechanisms. Both mechanisms require that the entire mesh is first sorted based on fitness value.




(a) Frontier Search [?]

The first mechanism, *adaptive clearing*, iterates through the sorted list and removes all nodes that are too close to this better node. The rationale is to prevent too many nodes searching the same space of the search area. The algorithm attempts to shift its focus from exploration towards exploitation by decreasing the adaptive clearing threshold at certain defined iterations. These relaxations of the adaptive clearing threshold are defined in Eq. 3.

Secondly, the algorithm will select the $N$ nodes with best fitness values from the resulting mesh. In case the size of the mesh falls below $N$ after adaptive clearing, the algorithm will randomly generate new nodes that will be added to the mesh.

### 2.3 Stopping Conditions

The algorithm will iterate through the expansion and contraction phases until a maximum number of fitness evaluations $C$ has been performed. After this, the node with best fitness is elected as final solution.

## 3 Research Questions

The paper aims to discover patterns that drive algorithm accuracy of VMO. Based on the description of the algorithm as provided in Section 2, the following statements will be analyzed and validated.

1. As explained in subsection 2.2, the algorithm discards nodes in each iteration without keeping a memory of failed exploration. Furthermore, node generation for local search (Eq. 1) and global search (Eq. 4), leaves the possibility of generating a new node at a fixed location, without random component.
   *How often will VMO explore the same location in the search area?*

2. Each pmh needs a healthy population of individuals to advance its search towards a better optimum. A new optimum can only be found if VMO generates nodes in more promising areas, thus preventing them from being removed by either adaptive clearing or the sort and select mechanism explained in subsection 2.2.
   *What is the efficiency of VMO in terms of node generation?*

3. The adaptive clearing mechanism tries to balance the exploration and exploitation capabilities of the algorithm. The original paper [**?**] compared the performance of VMO with and without adaptive clearing and concluded that the mechanism has a positive impact on the algorithm. However, no explanation has been given on the seemingly random moments at which the adaptive clearing threshold $\varsigma$ is relaxed, nor is the degree of each relaxation scientifically explained (See Eq. 3). As $\varsigma$ is used in local search (Eq. 1) and therefor relevant in both the expansion and the contraction phase, we assume it has a potential big impact on algorithm accuracy.

   (a) *What is the impact of an updated adaptive clearing threshold on the efficiency of VMO within an iteration?*

   (b) *What is the impact of an updated adaptive clearing threshold on the accuracy of VMO within an iteration?*

## 3.1 Definitions

Throughout the paper, the following definitions will be used.

- *Algorithm Accuracy*: The ability of the algorithm to find an accurate global optimum
- *Useful node*: A node that has been retained for at least one iteration, as opposed to nodes that are immediately removed. These are the nodes that are global or local bests or nodes that are used as parents for new node generations.
- *Algorithm Efficiency*: The ability of the algorithm to generate *useful nodes*
- *Adaptive Clearing Threshold* $\varsigma$: The minimum allowed distance that is used the adaptive clearing mechanism (See. Eq. 3). Nodes that fall within a distance $\varsigma$ of a better node, will be removed by adaptive clearing.
- *Relaxation of the adaptive clearing threshold*: The operation in which, at various points throughout the algorithm, the adaptive clearing threshold $\varsigma$ is decreased, thus allowing more areas around optima to be populated more densely.

# 4 Methodology

The analysis covers the *expansion* as well as the *contraction* phase of the algorithm. For the *expansion* phase, the main focus is on analyzing the exact location on which new nodes are generated. The *contraction* phase is evaluated by specifically monitoring the adaptive clearing mechanism and its impact on algorithm accuracy.

All algorithm runs have been performed with the configuration as depicted underneath. The values have been chosen in accordance with the original paper [**?**].

- Size of the mesh $N$: 20
- Maximum number of problem evaluations $C$: 100000
- Number of neighbors $k$: 3

The original paper is not unambiguous about the exact usage of adaptive clearing. We adopted the strictest interpretation in which a node is only removed if it falls within the minimum allowed distance $\varsigma$ for *each of the ten* dimensions.

The list of goal functions that has also been used in [**?**], is provided underneath. The functions have been evaluated in *ten* dimensions, with twenty independent runs executed per function. For each initialization of the mesh, it was ensured that the optimum would not be found within the boundaries of the mesh. This has also be done in the original paper, giving the algorithm a less straightforward path towards a solution. This is a common practice that has also been applied by Puris in the original paper [**?**] and aims at giving the algorithm a less straightforward path towards the solution. Both exploration and exploitation can be validated as VMO has to *move* its mesh of nodes across the search space rather than immediately exploiting the global optimum.

$f_1$ Sphere function

$$f_1(x) = \sum_{i=1}^{D} x_i^2 \tag{8}$$

Simple, convex function with a global optimum in 0.

$f_2$ Rosenbrock's function

$$f_2(x) = \sum_{i=1}^{D-1} (100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2) \tag{9}$$

Shaped like a through, with the global optimum in 0. Fitness values rapidly improve while entering the through, but only marginally decrease while searching within the through. Locations relatively far from the global optimum may still have a similar fitness value to the optimum.

$f_3$ Griewank's function

$$f_3(x) = \frac{1}{4000}\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1 \tag{10}$$

A general downward trend towards the global optimum in 0. The trend is modified by the cosinus function, creating a more hilly, but relatively smooth, function landscape.

$f_4$ Rastrigin's function

$$f_4(x) = \sum_{i=1}^{D}(x_i^2 - 10cos(2\pi x_i)) + 10D \tag{11}$$

Displays many local optima that are regularly distributed throughout the function landscape. Optimum is located in 0. Similar to $f_3$ but with more narrow peaks and throughs.

$f_5$ Weierstrass' function

$$\begin{aligned} f_5(x) = \sum_{i=1}^{D}(\sum_{k=0}^{20}(0.5^k \cos\left(2\pi 3^k(x_i + 0.5)\right))) \\ -D(\sum_{k=0}^{20}(0.5^k \cos\left(\pi 3^k\right))) \end{aligned} \tag{12}$$

Very irregular function that shows a general trend towards the global optimum in 0, but is never gradually declining nor increasing and displays constant upward and downward movements that is less smooth and more erratic than for example $f_3$.

$f_6$ Ackley's function

$$\begin{aligned} f_6(x) = -20exp(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2} \\ -exp(\frac{1}{D}\sum_{i=1}^{D} cos(2\pi x_i)) + 20 + exp(1) \end{aligned} \tag{13}$$

Cone-shaped function with a narrow, deep global optimum in 0.

Detailed algorithm execution is logged by storing each creation and removal of a node in a database. Apart from this individual node life cycle, we record each identification of a new optimum as well as the relaxations of the adaptive clearing threshold.

This data is used to examine the different locations in which nodes are generated. In addition, we correlate the number of new nodes as well as the convergence of the algorithm to the adaptive clearing mechanism.

# 5 Results

Table 1 provides an overview of the statistical results retrieved from the different runs on the goal functions. The first column gives an indication about *algorithm efficiency* and depicts both the absolute and relative amounts of *useful nodes* that the algorithm managed to produce on average. In general, between 1 and 22% of all nodes generated are retained for a next iteration. All other nodes are discarded immediately, within the very iteration they are created in.

Next in the table, *adaptive clearing* refers to the number of nodes that are removed by adaptive clearing. Note that adaptive clearing is by design only applied on nodes that are not considered *useful*. The percentages indicate that adaptive clearing is responsible for about half the removals that are performed by the algorithm.

*Revisits* refers to the number of times the algorithm generated a node on *exactly* the same location that has already been evaluated before. For $f_3$, a surprisingly high 36.64% of nodes are generated at a location where previous iterations already generated a node. Please be reminded that this is an *exact* location in *ten* dimensions where coordinates are displayed with a precision of 12 decimals.

The last column of Table 1 displays the average number of times the algorithm elected a new node as global best to replace the previous temporary solution. Out of the 100000 nodes generated, very few appear to be an improvement to the current best solution.

In figures 3, 4, 5, 6, 7, and 8, a detailed insight in VMO execution has been provided. The x-axis depicts the progress of the algorithm and is expressed in percentiles of node generation. With a maximum number of problem evaluations equal to 100000, each percentile roughly represents the creation of 1000 new nodes. The bar plot represents the number of *useful nodes* that are generated within that percentile and is expressed as the percentage of *useful nodes* in the total number of nodes generated in that percentile. The bar chart can be considered as the evolution of *algorithm efficiency*. In general, the efficiency shows a steady decline from the start but seems to display short peaks at the moments that correspond with a relaxation of the adaptive clearing threshold $\varsigma$.

The line plot within the same graphs can be seen as the evolution of *algorithm accuracy*. It represents the percentage improvement in the global optimum that has been achieved compared to the previous percentile and is calculated as in Eq. 14. This relative representation was chosen to display accuracy in the later stages of the algorithm, when improvements are typically small in absolute terms. Improvements of the optimum seem to be somewhat unpredictable, but clear trends are emerging anyway. Starting from a random population, the algorithm can easily find better solution. Over time, improvements become less likely and less dramatic, with the exception of peaks that again correspond to the relaxation of the adaptive clearing threshold $\varsigma$.

Table 2 provides statistical support for a correlation between a relaxation of the adaptive clearing threshold and algorithm accuracy. The table displays the

Table 1: Summary

| F | Useful Nodes | Adaptive Clearing | Total Revisits | Nr. New Optima |
|---|---|---|---|---|
| $f_1$ | 1615.85 | 46224.2 | 25303.45 | 79.95 |
| | 1.62% | 46.20% | 25.29% | |
| $f_2$ | 2080.25 | 49888.5 | 9045.95 | 106.70 |
| | 2.08% | 49.87% | 9.04% | |
| $f_3$ | 1270.35 | 46290.6 | 36659.55 | 70.35 |
| | 1.27% | 46.27% | 36.64% | |
| $f_4$ | 1374.00 | 47517.6 | 477.00 | 99.80 |
| | 1.37% | 47.50% | 0.48% | |
| $f_5$ | 1173.35 | 52178.6 | 27.35 | 66.45 |
| | 1.17% | 52.16% | 0.03% | |
| $f_6$ | 1435.6 | 46220.95 | 3291.2 | 83.6 |
| | 1.44% | 46.22% | 3.29% | |

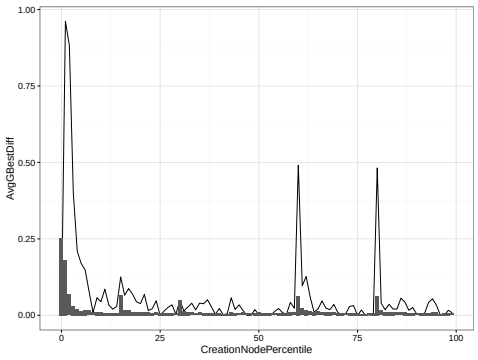



Figure 3: $f_1$ Node generation and optimum improvements

average relative improvement of the optimum within percentiles where the adaptive clearing threshold is relaxed ($\varsigma_0$) and compares it to the average relative improvements in percentiles with a constant threshold ($\varsigma_1$). Statistical output from a Welch t-test supports the hypothesis that a relaxation of adaptive clearing has an impact on the average optimum improvement at a significance level 0.05.

$$\delta_p = \frac{gbest_{p-1} - gbest_p}{gbest_{p-1}} \tag{14}$$

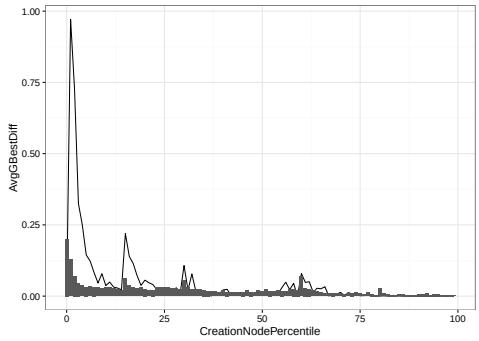



Figure 4: $f_2$ Node generation and optimum improvements

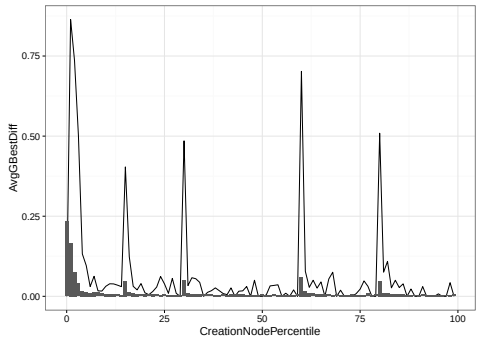



Figure 5: $f_3$ Node generation and optimum improvements

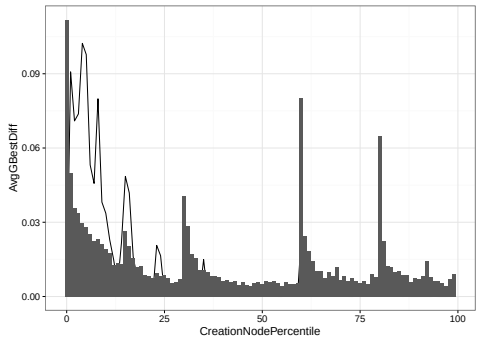



Figure 6: $f_4$ Node generation and optimum improvements

# 6 Discussion

By design, VMO's ability to improve its current optimum is influenced by the success rate of its node generation. Because only the most promising nodes are

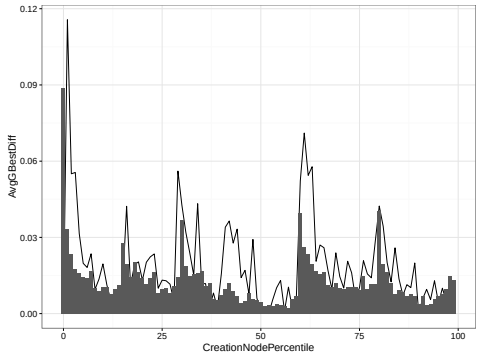



Figure 7: $f_5$ Node generation and optimum improvements

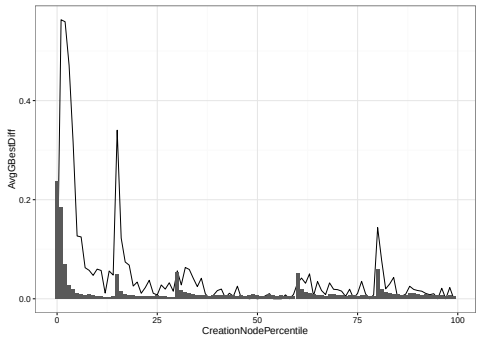



Figure 8: $f_6$ Node generation and optimum improvements

carried over to the next iteration, we can consider both the number of *useful nodes* and the average age of these nodes reasonable indicators for algorithm accuracy.

With between 98 and 99% of nodes immediately discarded, Table 1 indicates a low level of efficiency in node generation. The algorithms inability to generate useful nodes becomes further evident by looking at the high amount of nodes removed by adaptive clearing as well as the number of locations that are explored more than once. Ranging from anywhere between a lowly 0.03% and an astonishing 36.64%, the analysis indicates a major potential drawback of VMO. Due to the rigid placings of new nodes (See Eq. 4 and 1), the algorithm runs the risk of generating and removing nodes in exactly the same locations over and over again. Imagine a node generating a new node halfway towards a local best. If the new node is not retained for the next iteration, but the parent nodes are, the process between these nodes will loop eternally until adaptive clearing modifies the generation or removal rules or one of the parents is removed from the selection by chance.

In addition to the aforementioned problem, about half of the nodes that are

Table 2: Average optimum improvements in relation to adaptive clearing relaxation

| | Average Improvement | | Statistics | | |
|---|---|---|---|---|---|
| $f$ | $\xi_0$ | $\xi_1$ | t | df | p |
| $f_1$ | 27.01% | 2.89% | 7.8742 | 84.914 | 1.019e-11 |
| $f_2$ | 10.16% | 2.33% | 4.4679 | 80.62 | 2.549e-05 |
| $f_3$ | 50.00% | 2.53% | 17.159 | 84.248 | $<$ 2.2e-16 |
| $f_4$ | 1.85% | 0.56% | 2.1893 | 85.867 | 0.03129 |
| $f_5$ | 3.90% | 1.75% | 2.3209 | 76.576 | 0.02296 |
| $f_6$ | 12.72% | 2.41% | 5.8502 | 90.465 | 7.733e-08 |

generated by VMO, are placed in areas from which they are immediately to be removed by adaptive clearing.

The first two research questions, related to *useful nodes* and the exploration of the search area, cannot be answered favorably based on the results above. Due to the high level of subjectivity in what would constitute a healthy efficiency in node generation, it is not straightforward to add statistical evidence to these claim.

The Figures 3, 4, 5, 6, 7, and 8, and the statistical proof in Table 2, seems to suggest a significant correlation between a relaxation of the adaptive clearing threshold and the ability of the algorithm to find an improved optimum as well as the ability of the algorithm to produce more *useful nodes*.

The correlation between a relaxed adaptive clearing and the number of *useful nodes* that are generated, can be considered logical and expected. A smaller threshold should result in a lower number of nodes removed under adaptive clearing, hereby increasing the likelihood of survival for nodes that are generated close to a local or global optimum.

While the effect of adaptive clearing on the number of *useful nodes* was to be expected, the correlation between a relaxed adaptive clearing threshold and algorithm accuracy is not as intuitive as it may seem. As adaptive clearing does not influence the *amount* of generated nodes, the improved accuracy cannot be attributed to a higher statistical probability of finding a better solution. A better explanation can be found in the fact that the adaptive clearing threshold is taken into account during the expansion phase, thus influencing the locations of new nodes. Additionally, refraining the algorithm from removing nodes closer by a local or global optimum and instead using them as a basis to generate new nodes seems to have a positive influence on algorithm accuracy. One might suggest that nodes that are closer towards an optimum are more likely to find an improvement to that same optimum.

It is worth pointing out that the data seems to suggest that the effect of adaptive clearing relaxations is short-lived. After each relaxation, optimum improvements become less likely. The algorithm soon needs another shock starter in the form of a new adaptive clearing relaxation in order to keep on finding

new optima. This again may be an indication that the node generation rules can be improved.

This claim is supported by the number of nodes that are generated in a previously explored location. Again, we refer to the inclination of VMO to rigidly generate nodes at an exact location (see Eq. 1).

# 7 Conclusion

This paper focuses on discovering internal patterns of VMO that drive algorithm performance. We introduced the term *useful nodes* to indicate all nodes that are carried into a next iteration and thus used as a basis to create new nodes. Next to the age of nodes, the ratio of the *useful nodes* in the total number of nodes can be used as an indicator for algorithm efficiency.

Regarding *algorithm efficiency*, the results seem to suggest that VMO only uses the information from a very limited set of nodes, compared to the total number of nodes generated. Firstly, the number of *useful nodes* generated by the algorithm is relatively low (between 1 and 2% of total nodes generated). Secondly, a potentially high number of locations is visited multiple times. Both indicators suggest the inability of VMO to produce nodes in promising areas of the search space.

Focusing on accuracy, a significant correlation between optimum improvements and the adaptive clearing mechanism seems to exist. Each relaxation of the adaptive clearing threshold seems to provide a short-lived boost to the algorithm's ability to improve on its current solution.

In general, the results seem to suggest that VMO's expansion phase could be improved by modifying node generation rules so that the resulting node locations will have a higher degree of randomness. Alternatively, or additionally, the algorithm could benefit from a memory that keeps track of locations already explored. As it is currently the case, VMO does not use the information provided by the nodes that are removed because of their insufficient fitness value or proximity to a better solution.

The results also lend proof to revise the contraction phase. Adaptive clearing seems to have a beneficial impact on algorithm accuracy. Modifications of the adaptive clearing mechanisms seem to be very likely to increase algorithm performance. A well tuned relaxation of the threshold $\varsigma$ could optimize the number of *useful nodes* generated by VMO, while simultaneously boosting its accuracy. Future research could focus on selecting the ideal iterations in which to relax the adaptive clearing threshold, while also determining an optimal degree of relaxation. The degree of the relaxation will likely be characterized by the balancing act between problem exploration and problem exploitation, a common feature in pmh-research. The original implementation does not propose to parameterize the adaptive clearing mechanism. Future research could delve into this to suggest alternatives, potentially allowing the algorithm to self-adapt it clearing mechanism to the problem set.

Based on the very low efficiency and a potential sub-optima adaptive clearing

algorithm, it seems very impressive that the VMO is nevertheless able to produce competitive results. Given this competitiveness, the results seem to lend credit to the idea of VMO but suggest that the implementation can still be improved. For future research, it would therefore be very interesting to tackle the problems that are mentioned in this paper and try to produce a superior algorithm.