mathematics

MDPI

Article

Niching Multimodal Landscapes Faster Yet Effectively: VMO and HillVallEA Benefit Together

Ricardo Navarro * and Chyon Hae Kim

Faculty of Science and Engineering, Iwate University, Ueda 4-3-5, Morioka, Iwate 020-0066, Japan
* Correspondence: ricardo@iwate-u.ac.jp

Received: 24 March 2020; Accepted: 20 April 2020; Published: 27 April 2020

Abstract: Variable Mesh Optimization with Niching (VMO-N) is a framework for multimodal problems (those with multiple optima at several search subspaces). Its only two instances are restricted though. Being a potent multimodal optimizer, the Hill-Valley Evolutionary Algorithm (HillVallEA) uses large populations that prolong its execution. This study strives to revise VMO-N, to contrast it with related approaches, to instantiate it effectively, to get HillVallEA faster, and to indicate methods (previous or new) for practical use. We hypothesize that extra pre-niching search in HillVallEA may reduce the overall population, and that if such a diminution is substantial, it runs more rapidly but effective. After refining VMO-N, we bring out a new case of it, dubbed Hill-Valley-Clustering-based VMO (HVcMO), which also extends HillVallEA. Results show it as the first competitive variant of VMO-N, also on top of the VMO-based niching strategies. Regarding the number of optima found, HVcMO performs statistically similar to the last HillVallEA version. However, it comes with a pivotal benefit for HillVallEA: a severe reduction of the population, which leads to an estimated drastic speed-up when the volume of the search space is in a certain range.

Keywords: AMaLGaM; clustering; estimation of distribution; evolutionary algorithm; framework; heuristic; hill-valley; multimodal optimization; niching; variable mesh optimization

1. Introduction

In the arena of optimization, a heuristic algorithm seeks for solutions that are good enough (not necessarily optimal) within a fair computation time [1]. Thus, it is crucial to keep the balance between the quality of the approximated solutions and the time used to reach them. Beyond simple heuristics for specific problems, metaheuristics are intelligent mechanisms that guide other heuristics through the search process [2]. Evolutionary algorithms (EAs) are a category of metaheuristics based on biological evolution. As global optimization methods, typical single objective EAs seek for a unique global optimum, ignoring the possible existence of other optima. That means a serious limitation in industrial scenarios described by multimodal optimization problems, such as those reported in [3,4]. In this context, the term multimodality denotes the presence of optima in various regions of the search space. Many decisions in engineering, e.g., selecting a final design, depend on the earlier optimization of relevant aspects, i.e., cost and simplicity [5]. Hence, experts look for several optima instead of the only best solution, so they can choose the most suitable by considering further practical aims.

In the course of the last four decades, researchers have steadily coped with that dilemma, leading to the resultant field of evolutionary multimodal optimization [6]. The simultaneous detection of multiple optima in the search space is a big challenge. Such a high difficulty, together with the vast range of application domains comprised, make it a very active research field that intends the devise of niching algorithms [6,7], which are the key methods here, and their coupling with metaheuristics. Inspired by the dynamics of the ecosystems in nature, the paradigm of niching is the common computational choice for multimodal optimization. Niching techniques have complemented several metaheuristic

approaches, e.g., Genetic Algorithms (GAs) [5–10], Particle Swarm Optimization (PSO) [11–14] and Differential Evolution (DE) [15–18]. Because of their strong synergy, niching methods have also been explained as the extension of EAs to multimodal scenarios [19].

The Variable Mesh Optimization (VMO) [20] metaheuristic has proved to perform competitively in continuous landscapes. In its canonical form, it is capable to locate different optimal solutions but it cannot maintain them over the time [21]. A few works [21–23] have augmented VMO as to make it cope with multimodality. They include VMO-N [22], a generic VMO framework for multimodal problems, whose only two instances are termed Niche-Clearing-based VMO (NC-VMO) [21] and Niche-based VMO via Adaptive Species Discovery (VMO-ASD) [22]. The experimental analysis demonstrated that VMO-N is a suitable approach for multimodal optimization. However, a final request in [22] persists as the search for a strongly competitive variant of VMO-N, which uses a more robust niching procedure than those in NC-VMO and VMO-ASD.

To annul that limitation, we unveil a new case of VMO-N that exploits the Hill-Valley Clustering (HVC) niching technique [24], and the Adapted Maximum-Likelihood Gaussian Model Iterated Density-Estimation Evolutionary Algorithm with univariate Gaussian distribution (AMaLGaM-Univariate) [25,26]. Recently, Maree et al. [24,27] successfully used those methods together in the HillVallEA scheme. Motivated by their relevant outcomes, we instantiate the VMO-N framework by incorporating such a joint strategy. The ensuing HVcMO method is expressly dubbed HVcMO20a when pointing at its primary setup. This is the first competitive version of VMO-N. To support this claim, we compared it to remarkable metaheuristic strategies for multimodal optimization. Further than the VMO-N's instances, HVcMO20a overcomes the Niching VMO (NVMO) [23] method, as far as we know, the other VMO-based approach for multimodal optimization reported in the literature.

In addition, HVcMO is not only a case of VMO-N, but also an extension of HillVallEA19, the ultimate version of HillVallEA, a quite sophisticated metaheuristic for multimodal optimization. HillVallEA19 [27] outperforms the algorithms presented in the last two editions (2018 and 2919) of the Competition on Niching Methods for Multimodal Optimization, within the Genetic and Evolutionary Computation Conference (GECCO) [28]. In spite of that, a strong weakness of HillVallEA19 is that it tends to need very large populations. However, it is not occasional at all that an effective multimodal optimizer suffers such a drawback, considering that many functions have a large number of optima to be found, and that many of them involve numerous variables as well.

As expected, observed results show that the combined use of HVC and AMaLGaM-Univariate within HVcMO make that VMO-oriented optimizer capable to approximate multimodal problems effectively. On the other hand, the application of the search operators of VMO allows such an extended HillVallEA mechanism to perform faster over a large set of problems among those used in this study, which derives from an overall reduction of the population size on the problem in the test suite. This fact evidences the mutual benefit of using both approaches together. Given new multimodal optimization problems, apart from those seen in this study, it is then possible to recommend either HVcMO20a or HillVallEA19 to solve such problems, according to their common characteristics with the benchmark functions approximated by these algorithms.

Multimodal optimization is theorized in Section 2, together with some niching-related concepts. Relevant works, including VMO-N and HillVallEA, are reviewed in Section 3, where some ideas to deal with outlined shortcoming are exposed as the objectives of this research. The VMO-N framework is improved in Section 4, and then instantiated as the HVcMO algorithm. Section 5 describes the setup for the experiments to validate and analyze such a new proposal, whose results are discussed in Section 6. Finally, Section 7 comes with the conclusions and some directions for future work.

2. Formal Notion of Multimodal Optimization and Niching Approach

For a sake of simplicity, multimodal optimization is usually defined in informal manners, mostly by means of some descriptive cases of multimodal scenarios, like in Figure 1. Such conceptualizations are definitely valid and assure a straightforward understanding of what multimodal optimization is.

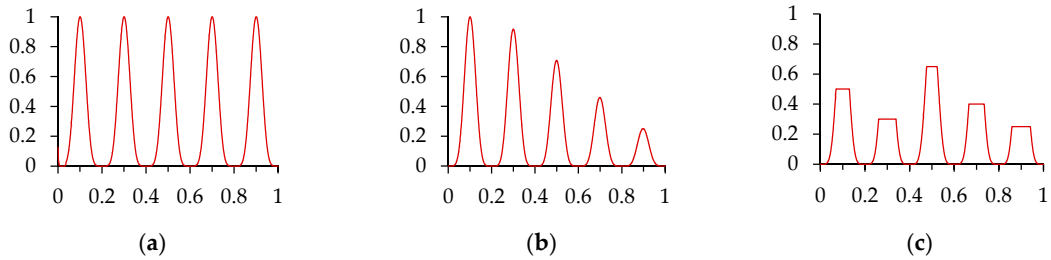

Figure 1. Examples of multimodal maximization functions, including a couple of usual benchmarks: (**a**) Equal Maxima; (**b**) Uneven Decreasing Maxima. The last graph illustrates: (**c**) a case with plateaus.

However, even supported by examples of functions, a more formal viewpoint is necessary. Let us formulate a continuous optimization problem P as a model driven by the elements below:

$$P \leftarrow (S, \Omega, f) \tag{1}$$

standing S for a search space, an abstract construction of all the possible solutions over a finite set of decision variables X_j, with $1 \leq j \leq D$, where D is the dimension of P. For each X_j, the pair of lower (a_j) and upper (b_j) bounds (limits) of its domain is given in $B \leftarrow \left\{ \left[a_j, b_j \right] \middle| 1 \leq j \leq D \right\}$. Being Ω the set of constraints between variables, P is an unconstrained problem if $\Omega = \{\}$. Finally, $f : B \rightarrow \Re$ indicates the objective function to optimize (either minimize or maximize).

In this scenario, instantiating a variable X_j means to assign a real value $v_j \in \left[a_j, b_j \right]$ to it, that is:

$$X_j \leftarrow v_j \tag{2}$$

and therefore, a solution $s \in S$ is a complete assignation where the values given to the decision variables satisfy the constraints in Ω. A solution s^* signifies a global optimum for a minimization problem P if and only if f reaches its lowest value at s^*, among all the solutions in S. Contrarily, when P is a maximization problem, s^* is said a global optimum iif $f(s^*)$ is the largest value of the objective function in S. Those two criteria for global optimality can be defined in a formal fashion as:

$$\exists (s^*)[GMin(s^*, P) \Leftrightarrow s^* \in S \wedge \forall (s)[s \in S \Rightarrow f(s^*) \leq f(s)]] \tag{3}$$

$$\exists (s^*)[GMax(s^*, P) \Leftrightarrow s^* \in S \wedge \forall (s)[s \in S \Rightarrow f(s^*) \geq f(s)]] \tag{4}$$

Likewise, a solution s' may be a local optimum for P, i.e., in a certain region $S' \subseteq S$. Since the decision variables are real-valued, the number of subsets of S is infinite, and seemingly the number of local optima, but only the subspaces that represent *peaks* count. The set of peaks (Pk) denotes the partition of S whose elements are regions where f is quasi-convex, for a minimization task:

$$\exists (Pk)[Peaks(Pk, P) \Leftrightarrow Partition(Pk, S) \wedge \forall (S')[S' \in Pk \Rightarrow QuasiconvexInRegion(f, S')]] \tag{5}$$

or quasi-concave, in the event of a maximization problem:

$$\exists (Pk)[Peaks(Pk, P) \Leftrightarrow Partition(Pk, S) \wedge \forall (S')[S' \in Pk \Rightarrow QuasiconcaveInRegion(f, S')]] \tag{6}$$

In Figure 1, each example of maximization consists of five (quasi-concave) peaks. In the rightmost graph, the peaks indeed end in 'plateaus', i.e., flat regions of S where the infinite encircled solutions have exactly the same fitness value. Formally, the local optimality can be defined as:

$$\exists (s')[LMin(s', P) \Leftrightarrow \exists (S')[S' \in Pk \wedge s' \in S' \wedge \forall (s)[s \in S' \Rightarrow f(s') \leq f(s)]]] \tag{7}$$

$$\exists (s')[LMax(s', P) \Leftrightarrow \exists (S')[S' \in Pk \wedge s' \in S' \wedge \forall (s)[s \in S' \Rightarrow f(s') \geq f(s)]]] \tag{8}$$

While a global optimum is also a local one in the peak it belongs to, it is common to disjoint those concepts for practical reasons. Let us S'^* denote the (complete) set of local optima (i.e., one optimum per peak; infinite if a plateau), and S^* be the set of global optima, so that $S^* \subseteq S'^* \subseteq S$. Hereinafter, by 'local optima' we refer only to the partial set of local optima (S^{**}), which excludes the global ones:

$$S^{**} \leftarrow S'^* - S^* \tag{9}$$

The global optimization approach solely accepts that $S^* = 1$ and $|Pk| = 1$. Diametrically, the multimodal optimization paradigm concerns the existence of multiple peaks ($|Pk| \geq 2$), not only several optima. For instance, a single-peak function ending in a flat region (infinite optima) is *weak unimodal* [29]. Solving a multimodal optimization problem means locating all peaks. The standard population-based optimizers are then modified to create stable subpopulations (known as *niches*) at all peaks to discover. Actually, niching algorithms exist for that purpose. Being the best solution in a niche referred as its *master*, every niche corresponds to a single peak of the multimodal problem, and most functions are featured by peaks of distinct radii, usually unknown. Several of those problems have been standardized as test cases for new multimodal optimizers. For some of them, the number of local optima is known precisely; in others, it is simply huge. Thus, only the global optima located are used for assessing the performance of new algorithms, which then might avoid efforts on finding local optima. It does not mean any shortage as most of those algorithms can be easily adapted to pursue global and local optima all together, when required in real-world scenarios.

3. Related Works

This section exposes key drawbacks of the multimodal optimizers. Besides, it elaborates on the basics of VMO and HillVallEA, the direct precedents of this paper. Despite VMO-N and NVMO are the two main augmented versions of VMO for multimodal scenarios, no study has addressed their conceptual differences. Such a lack directly calls for comparing them from theoretical viewpoints, which turns up as the first goal of this investigation. Some downsides of VMO-N and HillVallEA are underlined as well, together with other consequential research objectives to mitigate such limitations.

3.1. Evolutionary Multimodal Optimizers

Because of their population-based nature, standard evolutionary algorithms can locate several globally best solutions at the same time, instead of only one. There is no guarantee that they are kept in the population during the next iterations though. A simple alternative is to seed all those solutions in the population for the upcoming generation. However, it might cause some undesired effects, e.g., guiding the algorithm to previously explored regions of the search space and therefore delaying the overall search process. Another option is to 'memorize' the set of such globally best solutions, aside from the population. Some of them must probably belong to the same peak of the fitness landscape. Even so, by having an updated list of such fittest solutions, some looked-for optima might appear, eventually. While the literature reports several examples [18,20,24,30], every single EA may be adapted in line with that strategy, as to make it able to deal with multimodality. The efficacy of such augmented algorithms over multimodal problems will still rely on their search capability though.

On the contrary, niching methods represent the choice to turn EAs into multimodal optimizers whose success does not rely on their search abilities. By complementing them with niching methods, EAs are able to form different subpopulations and maintain them along the search process, in order to identify multiple optima together. As indicated in [21,22], in spite of the numerous works completed about multimodal optimization, it continues as a challenging research field due to various shortcomings of the niching algorithms. Certainly, their dependence on parameters that characterize the target functions, e.g., the radius of the niches, is the greatest weakness of multimodal optimizers. Furthermore, most of those optimization techniques are incapable to solve multimodal problems that have a large number of optima, which implies the necessity for more effective niching methods.

Evolutionary multimodal optimizers are also susceptible to the dimension of the problem, i.e., the number of variables involved. It is said that an optimization algorithm is scalable if it continues performing effectively when the dimension of the problem increases. Regarding this matter, Kronfeld and Zell [31] drew attention to the lack of studies about the scalability of multimodal optimization methods, a matter that remains. Other important drawbacks that have been poorly approached concern the high computational complexity of such techniques, which determines the time they need to approximate multimodal problems. This paper covers that issue, not in a generic manner but addressing only a couple of methods relevant for this investigation, namely the inspiring HillVallEA19 and our proposed extension of it, the HVcMO20a algorithm (also a case of VMO-N).

3.2. The VMO Metaheuristic

In the Variable Mesh Optimization scheme, the candidate solutions are denoted as nodes. Thus, the population is seen as a mesh that initially has I nodes $(s_1, s_2, \ldots, s_I)$. Every single node s_i, with $1 \leq i \leq I$, is represented as a real-valued vector $\left(v_1^i, v_2^i, \ldots, v_j^i, \ldots, v_D^i\right)$ of dimension D, i.e., $1 \leq j \leq D$. As described in Algorithm 1, the evolutionary search loop of VMO is controlled by four basic parameters, namely the size (I) of the initial mesh in each generation, the maximum number (T) of individuals in the total (expanded) mesh, the number (k) of nodes in the neighborhood of any given node, and the stop criterion (c). Note that in this paper, every collection starts with index 1.

The search process by VMO mainly consists of two stages: the mesh expansion and the mesh contraction, set together to keep exploration and exploitation in balance. The expansion occurs by dint of the main ways of search in VMO, namely the F-operator, the G-operator and the H-operator [20]. They respectively deal with the creation of new nodes toward the local optima (Algorithm 1, line 11), headed for the global optimum (line 15), and from those nodes in the frontiers of the mesh (line 20). Once the mesh is expanded, the contraction stage runs an elitist selection of nodes in the current generation to keep them in the initial mesh of the next iteration. Such a selection is previously affected by the adaptive clearing operator.

3.3. Is the VMO's Adaptive Clearing a Niching Method?

This technique is proved useful for VMO to deal with complex functions [32]. Despite that study concerns global optimization only, it fairly takes multimodal problems for benchmarking because of their high difficulty. When it comes to the research about VMO for multimodal optimization [21–23], the adaptive clearing has shown controversial though. To analyze about that, it is beneficial to have look at a certainly similar niching method, the classic clearing proposed by Petrowski [33]. Beyond the common aspects, a few relevant facts make them divergent, as the descriptions in Algorithm 2 suggest. In this context, for the sake of parity, the population handled by the Petrowski's method is referred as M', which actually occurs when it is picked as the niching idea in VMO-N, like in [21].

The typical clearing depends on a parameter called niche radius that describes the peaks of the fitness landscape, whose value is often unknown. Another parameter, the number of winners, defines the quantity of solutions that may form a niche. After sorting all solutions by fitness in a decreasing order, they are sequentially analyzed to split the population into niches in line with these rules: (1) if no niche is yet created or the maximum possible distance (i.e., the radius) to the master of the current niche is reached, the next fittest solution is marked as the master of a new niche; (2) the subsequent solutions belong to the last created niche only if its spatial distance to its master does not exceed the niche radius; and (3) among those individuals that belong to the same niche, only a pre-set number of them are actually added to the niche as winners, while the remaining ones are marked by setting their fitness to zero. The algorithm returns in full the collection of niches that eventually emerge.

Algorithm 1 Variable Mesh Optimization

Main phases: ■ mesh expansion ■ mesh contraction

Inputs: D-dimensional problem $P \leftarrow (S, \Omega, f)$ as in Section 2, including the domain bounds B, plus:

I: size of initial mesh — T: size of expanded mesh

k: number of neighbors — c: stop criterion

Process:

1. $M \leftarrow random\_nodes(I)$ // Initial mesh of I nodes
2. **repeat**
3. $s^* \leftarrow find\_fittest(M)$ // Globally best node
4. $M' \leftarrow M$ // Expanding mesh M'
5. $\varepsilon \leftarrow distance\_threshold(B, Time_{notion})$ // That notion refers e.g. a certain budget
6. **for** $i = 1$ **to** I **do**
7. $Lk_i \leftarrow find\_nearest\_nodes(k, s_i, M)$ // By spatial location
8. $S_i' \leftarrow \{s_i\} \cup Lk_i$
9. $s_i' \leftarrow find\_fittest(S_i')$ // Best node in the vicinity S_i'
10. **if** $s_i \neq s_i'$ **then**
11. $s_i^F \leftarrow create\_locally(s_i, s_i', \varepsilon)$
12. $M' \leftarrow M' \cup \{s_i^F\}$ // Local expansion
13. **end if**
14. **if** $s_i \neq s^*$ **then**
15. $s_i^G \leftarrow create\_globally(s_i, s^*)$
16. $M' \leftarrow M' \cup \{s_i^G\}$ // Global expansion
17. **end if**
18. **end for**
19. $|Ls^H| \leftarrow min(T - |M'|, I/2)$ // Count for frontier expansion
20. $Ls^H \leftarrow create\_from\_frontiers(|Ls^H|, M)$ // List of $\lfloor |Ls^H|/2 \rfloor$ nodes created from // the interior frontier and $\lceil |Ls^H|/2 \rceil$ // from the exterior one
21. $M' \leftarrow M' \cup Ls^H$
22. $descending\_sort\_by\_fitness(M')$
23. $adaptive\_clearing(M', \varepsilon)$
24. $M \leftarrow elitist\_selection(I, M')$ // Replace M by the best nodes (up to I)
25. **if** $|M| < I$ **then**
26. $Ls^{rand} \leftarrow random\_nodes(I - |M|)$
27. $M \leftarrow M \cup Ls^{rand}$ // Complete initial mesh for next cycle
28. **end if**
29. **until** c is reached

Output:

30. **return** s^*

Algorithm 2 Clearing methods: (**a**) Petrowski's clearing; (**b**) VMO's adaptive clearing

Inputs:
M': mesh expanded in a search space S, and B: set of domain bounds of the variables (Section 2)
σ: niche radius
κ: maximum number of winners (individuals in any single niche)
Process:
1. $Nh \leftarrow \{\}$
2. *descending_sort_by_fittest*(M')
3. **for** $i = 1$ **to** $|M'|$ **do**
4. **if** $fitness(s_i) \neq 0$ **then**
5. $winners = 1$
6. $niche \leftarrow \{\}$
7. $niche \leftarrow niche \cup \{s_i\}$
8. **for** $q = i + 1$ **to** $|M'|$ **do**
9. **if** $fitness\left(s_q\right) \neq 0$ **and** $distance\left(s_i, s_q\right) < \sigma$ **then**
10. **if** $winners < \kappa$ **then**
11. $winners = winners + 1$
12. $niche \leftarrow niche \cup \left\{s_q\right\}$
13. **else**
14. $fitness\left(s_q\right) \leftarrow 0$
15. **end if**
16. **end if**
17. **end for**
18. $Nh \leftarrow Nh \cup \{niche\}$
19. **end if**
20. **end for**
Output:
21. **return** Nh

(**a**)

Inputs:
M': mesh expanded in a search space S, and B: set of domain bounds of the variables (Section 2)
$Time_{notion}$: an estimate of the time passed, e.g. the budget used (given maximum number of either iterations or fitness evaluations)
Process:
1. $\varepsilon \leftarrow distance\_threshold(B, Time_{notion})$
2. *descending_sort_by_fittest*(M')
3. **for** $i = 1$ **to** $|M'|$ **do**
4. **for** $q = i + 1$ **to** $|M'|$ **do**
5. **if** $distance\left(s_i, s_q\right) < \varepsilon$ **then**
6. $M' \leftarrow M' - \left\{s_q\right\}$
7. $q \leftarrow q - 1$
8. **end if**
9. **end for**

Output:
10. **return** M'

(**b**)

Comparable to the method described above, the adaptive clearing is an important step in VMO. After sorting the nodes in the expanded mesh according to their fitness, the I best ones are meant to survive. However, the worst among those nodes closer (in fitness) than a distance threshold ε are removed. In that way, the clearing fosters diversity in the population, which is the reason it is planned in VMO for. Such a threshold, which is also used in the local expansion, is dynamically computed at each iteration, guaranteeing larger values at the beginning of the optimization process and smaller values at the end. Several studies [20–23] illustrate how to determine the threshold ε, which has a component ε_j for each j-th variable of the problem. Every single ε_j is computed by using the range $\left[a_j, b_j\right]$ that represents the domain of the of the variable X_j, together with the amount of budget wasted from total originally given. While other notions of budget are valid, the most commonly applied in VMO is the maximum number of either iterations or evaluations of the fitness function. In any case, ε_j means a portion of the amplitud between a_j and b_j; the less budget remains the shorter that portion is. The value of ε results from combining those of all its ε_j components, e.g., by averaging them.

In the original VMO, the adaptive clearing assures diversity among the individuals that form every initial mesh. Nonetheless, when used on multimodal optimization problems, such a clearing is also responsible for the loss of several globally fittest solutions found. In other words, it provokes the incapability of VMO to maintain the identified optima, a critical effect verified in [21]. Conversely, in [23] Molina et al. claimed to use the adaptive clearing as a niching method within NVMO, their

proposal for multimodal optimization, using the dynamic threshold ε as the niche radius. At this point, it is important to clarify a few facts about it.

It is deductible from their pseudocodes that same as the Petrowski's method, the adaptive clearing is capable to split the population considering the distance between the individuals. However, such division occurs in a nonconcrete way in the adaptive clearing since it does not create any explicit niche. Instead, it keeps in the population only those solutions that survive the clearing (same as in the typical VMO) as the masters of the abstract niches. While the typical clearing returns a collection of niches, each of which is formed by several winner solutions, the output of the adaptive clearing is nothing else but the contracted mesh, which consists of the masters only. That difference among both methods is relevant only if the adaptive clearing were used in any multimodal optimizer that takes into account not only the masters the niches in full. Even in such a hypothetical case, that issue is indeed easy to treat by making minor algorithmic modifications to the adaptive clearing procedure.

In its plain form, the adaptive clearing of VMO can be seen as a variant of the typical clearing, where (1) the niche radius is updated dynamically and (2) the number of winners is not taken into account, as (3) the masters are the only kept solutions in (4) the reduced population, instead of in explicit niches. Apart from the clearing procedure, other niching approaches that use the niche radius parameter, such as the classical fitness sharing [34] and the species conservation method [35], have been augmented with distinct strategies [36–38] to adapt the radius along the optimization process. It is logical to consider that their success strongly relies on the realism of the computed radius, for which two aspects are decisive: a good rule to modify its value and a control mechanism to stop such an update at any convenient moment. The latter is extremely hard to plan, although it is very desired. Irrespective of being increased or decreased, the updating value should reach the actual (unknown) radius, eventually. Further modifications of the niche radius will make the multimodal optimizer detect a wrongly short or large number of niches, and mayhap the recently identified ones will not appear again. A practical compensating option is to combine those revamped niching methods with other strategies within the multimodal optimizers, e.g., by saving aside the best solutions found, or by conditioning the stop of the overall optimization process to the number of the new niches found.

In VMO, the value of the threshold ε decreases during the search process, as explained above. However, the adaptive clearing does not include any strategy to attempt identifying when such a desired value of the niche radius is apparently reached. Instead, ε continues decreasing as the search process goes on. From the niching viewpoint, another issue is that ε is assumed the same for all the peaks in every function, which is not rare in those methods that use a niche radius parameter, making them inappropriate for those functions having several peaks of different radii. In addition, the evidence reported in [21] shows that those nodes approximating certain peaks of the analyzed benchmark problems are partially or totally 'cleared' from the mesh. For that reason, every time VMO jumps to a new iteration, for some of the peaks the search starts over. That happens during the whole optimization process, i.e., regardless of the value of ε. It is then impossible for VMO to maintain the fittest solutions. Then, how can NVMO succeed on dealing with multimodal optimization problems? And, what is the actual benefit of the adaptive clearing there? Section 3.4.1 answers both questions, while approaching the essentials of such a metaheuristic scheme.

3.4. VMO Niching Strategies

As it was originally proposed for global optimization, a few algorithmic proposals appeared to induce VMO to preserve (along the time) the multiple optima this method can find when solving multimodal problems (see Figure 2). They can be reduced to the following two approaches: the NVMO method and VMO-N, the generic niching framework for this metaheuristic. Those methods are described below in an overall fashion. Besides, considering that they are the two main adaptations of VMO to deal with multimodality, it is important to bring out a few differences between them, completing the first objective of this investigation.

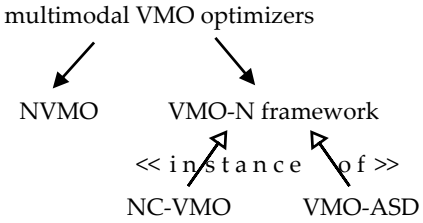

Figure 2. VMO for multimodal optimization.

3.4.1. Niching VMO

The details of NVMO were discussed by Molina et al. in [23], where authors highlighted the major augmentations to the standard formulation of VMO. They can be summarized as follows:

- With reference to the creation of new nodes from a global perspective, not only the fittest one in the population is considered for expanding the mesh. Instead, several globally best nodes are determined as those having a fitness value very similar to the best solution. Consequently, a new solution is created between every non-optimal node in the mesh and its nearest node (according to the Euclidean distance) among those marked as global optima.
- In order to improve the search capability of the metaheuristic, the Solis-West local search method [39] is run over only a certain number of nodes from the expanded mesh M'. The solutions selected are those located more distant from the globally fittest nodes found.
- The globally fittest nodes are kept in an external memory, whose update involves every $s_i \in M'$. If s_i has a 'visibly' better ($>10^{-6}$) fitness than the global optimum saved, the memory restarts with only s_i. Otherwise, s_i means a candidate global optimum if it is 'similar' ($\pm 10^{-6}$) in fitness to the global optimum. In that case, the memory accepts s_i if its closest global optima is located at a least distance denoted as the memory threshold (assumed in [23] as the accuracy level [40]).

NVMO does not split the population into niches; it accordingly executes the adaptive clearing over the whole mesh. The risk of deleting several global optima from the population persists. Thus, to keep the globally best nodes found, NVMO implements the aforementioned strategy comprising an auxiliary memory of such fittest solutions. In fact, the indubitable capacity of NVMO to deal with multimodal problems is due to the memory of optima, and not because of using the adaptive clearing with intention of niching. Such a memory is equivalent to others in previous and succeeding multimodal optimizers, e.g., the dynamic archive used in [18,30], and the elitist archive of HillVallEA.

Conforming to a common reasoning, other algorithms update the memory for the assumed global optima after executing the niching method, using the identified masters as the candidate optima. Those steps are run in the opposite order in NVMO, where such candidate solutions are selected from the entire expanded mesh, according to the explained optimality criterion, and the adaptive clearing occurs just after restructuring the memory. It means that the masters derived from that clearing procedure are not considered as the candidate optima to be kept in the external memory, a choice that might respond to a poor performance of the adaptive clearing as a niching method. Fostering diversity remains then as the main benefit of the adaptive clearing in NVMO, the same as in the original VMO. In addition, it is relevant the computational cost that the selection of the candidate optimal solutions represents in NVMO, since every single node in the mesh is processed for that, instead of considering only the masters of the identified niches.

As a multimodal method, NVMO does not rely on the niching step, but on the external memory. Thus, its effectiveness deeply counts on its search ability, strengthened by the Solis-West method. The use of local search was moved by a previous work [41] on PSO-based multimodal optimizers. In particular, NVMO runs the Solis-West procedure over a certain number of solutions in the expanding mesh, which is later completed with the new nodes resulting from that local search mechanism. After that, the list (memory) of globally best discovered solutions is updated, just before conducting the adaptive clearing over the mesh in order to prepare the population for the next generation.

3.4.2. VMO with Niching

Given in [22], the pseudocode corresponding to VMO-N suggests the moment where the niching method had better execute within VMO. In addition, it proposes to apply the adaptive clearing operator of VMO to each niche found, instead of over the whole extended mesh, as it occurs in the original form of such a metaheuristic. In that way, the efforts to assure diversity in the mesh do not affect the fittest solution (master) of each niche, which survives as part of the population to evolve. Consequently, VMO-N allows to maintain the discovered optima along the time. Furthermore, as a framework, it permits to include any desired niching method in its procedural workflow. Thus, the aforementioned instances of VMO-N are registered as NC-VMO and VMO-ASD.

The former method results from using the clearing operator by Petrowski (see Section 3.3) within VMO-N, as a niching mechanism. Obviously, it also applies the VMO's adaptive clearing (over every niche) as a diversity strategy. Being its first instance ever, the greatest impact of NC-VMO is to put VMO-N in practice. As the standard niching method of clearing depends on the value of the niche radius, NC-VMO suffers such a drawback too, making it suitable for a short number of multimodal problems. In response to that, VMO-ASD assimilates the Adaptive Species Discovery (ASD) niching method by Della Cioppa et al. [42], which does not rely on any parameter that describes the target fitness landscape. As a result, VMO-ASD empirically shows itself as a more extensive scheme than NC-VMO. None of them is strongly competitive though, compared to outstanding algorithms in the literature. That leads to the seek for a strong VMO-N variant, a need that is fulfilled in this study.

Before dealing with that shortcoming about the effectiveness of specific instances, the building blocks of VMO-N should be revised. As a framework, it should offer not only the possibility to properly incorporate any niching mechanism of preference (it can be decided freely), but also the chance to conduct further algorithmic strategies which might support the discovery of the niches and their maintenance along the time. In other words, VMO-N has to become a more flexible proposal by considering the inclusion of multiple (likely optional) procedural steps, which will also reinforce the generalizability of the framework.

3.4.3. Divergence between VMO-N and NVMO

For a better comprehension, Table 1 summarizes the most relevant dissimilarities between both approaches. One of the most obvious is the external memory used by NVMO to keep the set of global optima discovered. The application of the adaptive clearing either over the whole mesh or over each niche is a crucial difference here. However, the most significant contrast is given by the underlined algorithmic sequences. The implication of this matter is not trivial though. In Section 4, we revise the VMO-N framework. Consistent with that, future instantiations of it might use an external memory or avoid the action of the adaptive clearing, etc. Regardless of the procedural choices taken, no new case of VMO-N will ever conduct the niching process just after an external memory is updated. It does not contradict the possible modification of such a memory, followed by the application of the adaptive clearing but only as a strategy to sustain diversity in the population. Moreover, in that case it would still be run over every single niche, not over the entire mesh. Among other critical individualities, these last remarks derived from the analysis of their algorithmic constructions, sufficiently justifying that no future variant of VMO-N will match the NVMO method.

Table 1. Major differences between VMO-N and NVMO.

Criteria for Comparison	NVMO	VMO-N Framework
Algorithmic sequence (core actions in each iteration)	expansion → local search → memory update → adaptive clearing (as niching)	expansion → niching step → adaptive clearing (as for diversity)
Maintenance of the globally fittest found solutions	An extra memory is needed to keep such solutions	They are properly kept in the mesh (future variants may save them apart)
Formation of niches	The adaptive clearing is taken as a niching method but the masters are not used for multimodal purpose	Any niching method can be incorporated at will
Requirement of niching parameters	The computed threshold for the adaptive clearing is used as a dynamic niche radius	It varies, subject to the needs of the niching scheme, e.g., VMO-ASD is free of such parameters but NC-VMO uses a niche radius
Effecting of the adaptive clearing of VMO	It affects the whole mesh at once	It is executed, by separate, over each identified niche
Utilization of local search	The Solis-West method is used, in order to strengthen the search ability	No local search technique is applied (future instances may consider it)
Applicability	It can be applied to approximate a large range of multimodal problems	For each case, it relies mainly on the niching method, e.g., VMO-ASD is widely applicable; NC-VMO is not

3.5. Hill-Valley Evolutionary Algorithm

This method is closely related to the new version of VMO-N, proposed in this paper. Hence, it is important to recapitulate the fundamentals of the HillVallEA, whose main modules are the niching technique, the core search method, and the restart scheme with an elitist (external) archive. This subsection addresses such elements, before relating that EA from a general outlook.

3.5.1. Hill-Valley Function: A Pivotal Subject for HillVallEA

The hill-valley function (HVF) [43] is a mathematical abstraction acknowledged in the field of multimodal optimization and thus used by several niching approaches, e.g., the aforementioned ASD. As explained afterward, the HVF-based test is strategic in HillVallEA. The functioning of it concerns the quasi-convexity/quasi-concavity of f in particular subspaces. Let us the points s_i and s_q refer any pair of solutions ($i \neq q$) in an optimization landscape, like in Figure 3. To decide if such two solutions are not in the same niche, some test inner points are generated. If any single test point is poorer in fitness than both s_i and s_q, they are said to exist in different peaks of the objective function, i.e., they belong to separate niches. Thus, s_1 and s_4 are admitted to go in distinct peaks since at least s_2 has worse fitness than both of them. With less fitness than only one the target solutions, the test point s_3 cannot separate them. The result is then influenced by the number of inner points and their location.

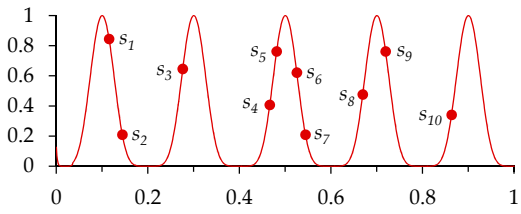

Figure 3. HVF test. It finds s_i and s_q in distinct niches if s_l ($i < l < q$) has less fitness than them.

Moreover, if none of the test points is found to have a worse fitness value than both s_i and s_q, it may be supposed that they belong to the same peak, but it cannot be affirmed. That occurs if s_5 and s_6 are the test points for s_4 and s_7, which are in the same peak; also if s_8 and s_9 are used as test points for s_7 and s_{10}, located in separate peaks. Hence, the only possible reliable conclusion by this test is that the two target solutions are in different peaks, if any of the inner points can prove it. Otherwise, the algorithms that split the population into niches via HVF, such as HVC, assume that those solutions are

in the same niche, on risk to fall in the false-positive case described above. Testing a large number of inner points is a logical intent to avoid that, but this is not feasible as it also increases the computational cost. Thus, proper mechanisms to complement the power of this test are required.

3.5.2. Hill-Valley Clustering: The Niching Method

By adopting the HVF-based test, the Hill-Valley Clustering algorithm shapes the niching plan for HillVallEA. The strategy of HVC to benefit from the HVF-based test is disclosed in [24]. That includes a routine to suggest the number of (equidistant) test points between two solutions, which is directly proportional to the Euclidean distance between such them. Besides, HVC implements a refined seek for the cluster that any given individual s_i belongs to, among those already identified. Given a selected population (*Sel*) sorted by fitness, excepting the initial solution, i.e., s_1, which is directly assigned to the first cluster created, each solution $s_i \in Sel$, with $i > 1$, is contrasted with up to $D + 1$ nearest fittest individuals via the HVF test. Among the solutions that exhibit a better fitness than s_i, it is tried with the closest one, and then with the second nearest, and then with the third one, and so on, until s_i is found to belong to the same cluster that some of such tested solutions. After $D + 1$ comparisons with negative result, s_i initializes another cluster.

Originally, that process was repeated for a total of $|Sel| - 1$ individuals, after which a resulting collection of clusters emerged as the found niches. Later [27], the authors concluded that it is worthy to contrast a given s_i with multiple top fittest neighbors only in two specific situations, which reduces the computation time to run the HVF-based test. The first scenario happens when s_i is one of the $\lceil |Sel|/2 \rceil$ solutions with better fitness. Otherwise, s_i is tested with the neighbor at hand only if they are separated by at least the expected edge length [24], which is the theorized distance of any pair of individuals equidistantly positioned in the search space. If it does not fall in any of those two cases, checking several fittest neighbors for s_i leads to discover accurate clusters but having a poor quality. Subsequently 'forgotten' by HillVallEA, they mean a waste of both the budget allotted to solve the problem and the computational effort. That principally concerns those problems with many local optima of very low fitness. Such a decision can be easily reconsidered in future versions of HillVallEA that intend to find not only the global optima, but the locally optimal solutions as well.

3.5.3. AMaLGaM-Univariate: The Core Search Method

HillVallEA is flexible about the incorporation of core search algorithms, as any can be freely included at no risk of interfering with any other fundament of such a scheme. Among other methods examined, AMaLGaM-Univariate performed stunningly within such a multimodal optimizer. That election resulted in the HillVallEA-AMu algorithm [24], renamed as HillVallEA18 and then refined as HillVallEA19, in [27]. The AMaLGaM-Univariate procedure is catalogued as an estimation of distribution algorithm (EDA) [44], a sort of EA suitable for those optimization tasks with a lack of knowledge about the objective function [26], e.g., the multimodal problems. Earlier studies [45–47] reported the use of EDAs to deal with multimodal optimization as well.

The EDAs seek for convergence by sampling a probability distribution updated all through the optimization process [26,44]. Hence, the focal action in every iteration is the estimation of such a distribution from the fittest individuals in the population. By using the distribution, a new generation of individuals is created to replace the formers, either fully or partly, after which only the best solutions remain in the recent population. Among the EDAs in the state-of-the-art on evolutionary computation, Bosman et al. [25] contributed the Adapted Maximum-Likelihood Gaussian Model Iterated Density-Estimation Evolutionary Algorithm, abbreviated first as AMaLGaM-IDEA and later as AMaLGaM. In [26], they analyzed three versions of AMaLGaM, basically differing on the Gaussian distribution used, being either fully multivariate, Bayesian factorized or univariately factorized. The last one led to AMaLGaM-Univariate, the succeeding core search method in HillVallEA.

The Gaussian probability distribution is conditioned by a vector of means (μ) and a covariance matrix (Σ). Given a cluster of solutions, that distribution is initialized in AMaLGaM-Univariate

by taking μ as the cluster means, and Σ as the covariance matrix of such individuals respecting μ. Then, a population with a size N_c is sampled, according to the initial distribution. In the each of next iterations, both μ and Σ are re-estimated and the population is later re-sampled. That repetitive procedure evolves to a success stage, after which the fittest solution is returned. Thus, the main parameters required by AMaLGaM-Univariate are the cluster per se (from which μ and Σ are initialized), and N_c. The later is coined as the 'cluster size' although it does not refer the number of individuals in the given cluster, but the size of the population used to optimize it. A value of tolerance is used to terminate AMaLGaM-Univariate when it is converging either to a presumed local optimum or towards an already assumed global optimum. The latest case is validated every five generations. Planning future variants of HillVallEA to find both global and local optima, requires one to modify such conditions to stop AMaLGaM-Univariate when converging again to the same optima (global or local). Nevertheless, the core search algorithm safely stops in other situations, e.g., when either the standard deviation of the solutions or the standard deviation of their fitness values is extremely small.

3.5.4. Elitist Archive, Restart Scheme and Overall Process

As to maintain the elites, i.e., the masters of the peaks, HillVallEA uses an archive E that in practice works alike the external memory in NVMO. Some facts differ in the mechanisms that control such structures though. For instance, the candidate optima (masters) to update the elitist archive are not picked by checking the entire population exhaustively. Other particularities of the strategy to update E towards the end of every evolutionary iteration are described below. The algorithmic details about HillVallEA addressed in this study combine from the two fundamental works [24,27], and the source code [48] that the authors unveiled under GNU General Public License v3.0. For their relevance to HVcMO (part of the proposal in Section 4), such details are simplified in Algorithm 3.

The evolutionary process is repeated while the remaining budget (function evaluations) is enough to at least generate other N individuals (Algorithm 3, line 5). In each iteration, a population of N individuals is randomly initialized according to a uniform distribution, and by applying a rejection strategy that came with HillVallEA19 (the latest version) to avoid re-exploring regions of the search space (line 6). Any new solution is very likely discarded (rejection probability $rp = 0.9$) if its nearest $D + 1$ solutions in the initial population of the prior generation were in the same cluster. Actually, $2N$ solutions are created with that rejection reasoning, but only N are chosen via a greedy scattered subset selection mechanism [49], to spread the initial population as it has proved to help the performance of EAs [27]. Once initialized, the population shrinks again, this time to a certain percentage indicated by the selection fraction τ: out of the N individuals, the fittest ones are taken. That seeks for higher outcomes by the overall optimization. The population is then made ready to be partitioned in niches. It is completed with those solutions stored in the elitist archive (line 10), so that they can work as attractors during the niching process, subsequently conducted via HVC (in line 11).

Being K the recent set of clusters, the core search method singly improves each $K_i (i \geq 1)$ whose best solution ($K_{i,1}$) does not match any former elite. Hence, the best solution in each final population by the core search method supposes a global optimum (line 15). The *Cnd* set of candidate optima is checked for updating E (in line 19). If the fittest candidate optimum exceeds the best elite in the archive, it is emptied. As well, those candidates poorer in fitness (by at least the given tolerance value) than the globally best solution, are labeled as local optima and then discarded. The others are assumed as global optima and included in E if they are new elites, i.e., they belong to different peaks than those already in the archive, which is checked by using HVF with five inner points. Otherwise, the presumed global optimum replaces the equivalent stored elite only when having a greater fitness value. In that way, this procedure intends to avoid cloning any saved optima.

In spite of the described efforts, in some cases no novel elite is detected. Authors ascribe that to a couple of possible reasons. The first one alludes to a population still insufficient to catch minor niches. The other supposition is that the number of individuals used by the search core method is inadequate to enhance complex niches. To cover both cases, the restart of the population for the next

cycle considers a larger number N of individuals, while the core search over each niche will also use an increased number N_c of solutions (see Algorithm 3, lines 21 and 22).

Algorithm 3 HillVallEA

Inputs: D-dimensional problem $P \leftarrow (S, \Omega, f)$ as in Section 2, plus the following main parameters:
$\mathcal{A}$: the core search algorithm (e.g. AMaLGaM-Univariate)
N: population size (in HillVallEA19: 64; in HillVallEA18: $16D$)
N^{inc}: increment factor for the population size (suggested value: 2.0)
$N_c^{\mathcal{A}}$: recommended population size for $\mathcal{A}$ (for AMaLGaM-Univariate: $10\sqrt{D}$)
N_c^{ini}: initial fraction of the population size of $\mathcal{A}$ (in HillVallEA19: 0.8; in HillVallEA18: 1.0)
N_c^{inc}: increment factor for population size of $\mathcal{A}$ (in HillVallEA19: 1.1; in HillVallEA18: 1.2)
$\tau_c^{\mathcal{A}}$: selection fraction for $\mathcal{A}$ (for AMaLGaM-Univariate: 0.35)
Tol: tolerance (by default: 1.0×10^{-5}; it may be set equal to the accuracy level)
MaxFE: budget expressed as the Maximum number of Function Evaluations

Process:

1. $N_c \leftarrow N_c^{ini} * N_c^{\mathcal{A}}$ // Actual population size for $\mathcal{A}$
2. $\tau \leftarrow \tau_c^{\mathcal{A}}$
3. $E \leftarrow \{\}$ // Elitist archive
4. $P_0 \leftarrow \{\}$
5. **while** $FE_{toSample}(N) \leq MaxFE - FE_{used}$ **do** // Enough FE to prepare N solutions?
6. $Pop \leftarrow sample\_uniform\_reject(N, D, P_0)$
7. $P_0 \leftarrow Pop$ // Population backup
8. $descending\_sort\_by\_fitness(Pop)$
9. $Sel \leftarrow truncation\_selection(Pop, \tau)$ // Portion of Pop said by τ
10. $Sel \leftarrow Sel \cup E$
11. $K \leftarrow HillValleyClustering(Sel)$
12. $Cnd \leftarrow \{\}$ // Set of candidate global optima
13. **for** $i = 1$ **to** $|K|$ **do**
14. **if** $K_{i,1} \notin E$ **then** // If the master of K_i is not a prior elite,
15. $cand \leftarrow \mathcal{A}(K_i, N_c, Tol)$ // run $\mathcal{A}$ to enhance that niche; take the
16. $Cnd \leftarrow Cnd \cup \{cand\}$ // fittest solution as a candidate optimum
17. **end if**
18. **end for**
19. $N_u \leftarrow update\_archive(E, Cnd, Tol)$ // Count elites either replaced or added
20. **if** $N_u = 0$ **then**
21. $N \leftarrow N * N^{inc}$ // If E remains unaltered, increase
22. $N_c \leftarrow N_c * N_c^{inc}$ // both population sizes
23. **end if**
24. **end while**

Output:

25. **return** E

A decisive matter to solve any optimization problem by using metaheuristics is the number of solutions required for it. Looking for multiple optima presupposes the demand for more individuals. Besides, some objective functions can be approximated with less solutions than others. Tuning the population size is definitely a challenge. In HillVallEA, such an amount is relatively short at first, and increments upon analysis. Besides, the initial size of the population changed from $16D$, in HillVallEA18, to 64 individuals in HillVallEA19. That signifies a smaller number of initial solutions for all problems with $D > 4$. In addition, as shown in Algorithm 3, HillVallEA19 uses lower values for both the initial fraction N_c^{ini} of the population of the local search method, and the factor N_c^{inc} that controls its increment. In spite of that, our observations (documented in Section 6) show that this method continues employing quite large populations to approximate well most problems at hand. This important

drawback markedly prolongs the running time, which drafts the next research goal of this paper, i.e., to lower such an execution time without any serious loss of effectiveness.

Because when it comes to metaheuristics, it is key to monitor the time, our concern about it is in consonance with a few algorithmic conceptions of HillVallEA. That includes the aforesaid adjustment of the parameters related to the population size. As well, to effect Hill-Valley Clustering over the set *Sel* of selected solutions, for each analyzed individual, the computed distances to all the solutions better than it are stored. Such a decision makes HVC more efficient, as its complexity noticeably drops from $O\left(D|Sel|^2\right)$ to $O(|Sel|)$. In addition, while avoiding re-examine areas of the search space, the restart scheme with rejection reduces time. That benefit is accentuated by using the subset selection method referred above, among other possible techniques that become inefficient when either the dimension of the problem or the sample size is large [27].

3.6. Research Objectives

The early declared research goal is actually attained above. Even so, just for correctness, it is recaptured as:

(1) To compare VMO-N and NVMO from a conceptual perspective

The previous examination of both VMO-N and HillVallEA put emphasis on their shortcomings. The main needs for research derived from those limitations determine most of the objectives of this investigation, recapped below in a clear manner:

(2) To revise the VMO-N framework
(3) To create a competitive variant of VMO-N
(4) To decrease the running time of HillVallEA while keeping it effective

Apart from those, a last need appears from a practical perspective. By confirming which of the relevant methods (previous and newly proposed) performs better on dissimilar test problems, it is easier to choose between them to approximate further (likely real-world) problems, having common features with the ones used here for benchmarking. In simple words, we pretend to answer the following question: is there any range of multimodal optimization problems for which one or the other analyzed method are preferred? That leads to the final research objective of this work:

(5) To find guidelines to select between past and new methods to better solve additional problems

This requirement derives from the well-known 'no free lunch' theorem [50], which supports that no metaheuristic absolutely outperforms all the others. Thus, none of them can solve any particular problem better than every remaining heuristic optimizer, irrespective of the subfield of optimization affected. It is then useful to have evidence for selecting methods to deal with upcoming problems.

4. The Proposals

In response to the second goal projected, this section presents the VMO-N framework with several improvements that guarantee a high flexibility. This new scheme constitutes the base for creating future VMO multimodal optimizers, such as the HVcMO algorithm, also introduced below.

4.1. Variable Mesh Optimization with Niching: A Revised Framework

This fresh proposal preserves the essential contributions of VMO-N: (1) advising when to better apply the niching step within VMO, and (2) applying the adaptive clearing operator over each niche. However, it comes with a strong adaptability by reason of the multiple optional commands, as Algorithm 4 explains. One of them is the use of an elitist archive *E*, a memory-based strategy imported from the literature. It has shown helpful in many multimodal optimizers [18,23,24,30,37], some of which were previously approached in this paper. Apart from the masters of the peaks, it might

be helpful to store other nodes with additional purposes. Hence, it is optionally considered a second external list, indicated as R in the framework.

The previous works about VMO-N overcame the initial shortcoming of VMO related to its incapacity to maintain the fittest found nodes in the population along the time. Given that, memorizing such nodes is unnecessary, unless certain circumstances apply, e.g., if they are required for implementing any mechanism, like the 'tabu' list in [37], to explicitly avoid re-visiting regions of the search space. However, this revamped VMO-N considers the possibility that the best individuals discovered are deliberately excluded from the mesh for the next iteration, either temporarily or permanently. In that situation, E is extremely needed.

Other optional instructions are carefully placed in the building blocks of VMO-N. Besides providing adaptability to that multimodal optimization scheme, they make it more generalizable, which means that its instances can deal with a vast amount of multimodal problems. A plethora of methods can be created by instantiating this framework in future. Actually, some of those non-compulsory commands indirectly suggest further strategies to apply. In addition, the number of parameters used and of variables declared is as large as needed, for example, by the niching algorithm incorporated or by any elective step conducted.

Based on the outcomes in [23,24], an important increase is the possibility of a local optimization step (Algorithm 4, line 31). In order to enhance the search, any freely chosen optimizer is separately run over each niche fund. It is said a local optimizer as it initializes from the individuals in a given niche, using either some of them or all. However, it might reach solutions in other areas of the search space, beyond the limits of the niche at hand. Following the main course of VMO-N, once it has been improved, the adaptive clearing affects each niche (in line 33) and then, the fittest node in the niche (indicated as $Nh_{j,1}$) is considered to update the list of elites only if $Nh_{j,1}$ was discovered just now. Moreover, in case of using the second external list (R), it is updated at that moment as well.

There are a couple of alternative courses involving the update of the memories (line 34), taking into account that it can be permuted either with line 33 or with line 35. If the order of the instructions in lines 33 and 34 are exchanged, the adaptive clearing occurs after updating the lists. In fact, that possibility was announced in Section 3.4.3, while explaining that even in that case the algorithmic sequence of any instance of VMO-N will differ from that of the NVMO algorithm. Such a modification has sense if, for example, several nodes have to be saved in the second extra memory before they are removed by the adaptive clearing operator.

On the other hand, permuting lines 34 and 35 indicates that the memory update happens at once, after processing all niches, instead of after processing every single niche. In that case, the set $\{Nh_{j,1}, \ldots\}$ contains the master of each j-th niche to update E, together with other relevant nodes to update R. What is more, it is possible to update one of the lists (E or R) inside the *for*-loop, every time a niche is processed, and the other list after that, only once. That alternative derives from the separability of both lists, and from the permutability of line 34. Nonetheless, effecting the adaptive cleaning (line 33) is no longer obligatory, because other mechanisms may satisfy its main function: to foster diversity in the mesh.

The VMO-N framework also integrates the notion of what we coin as *global key nodes*. That implies not only the single fittest solution but any other with global importance, for example, every node having similar fitness than the best solution in the population. Whenever that option is taken, the global expansion of the mesh takes into account the set Gk^* of such key solutions. Likewise, the local expansion is slightly adapted. Given any s_i node, its k neighbors are not necessarily selected from the whole mesh. Instead, they are found among those nodes belonging to a certain universe that is specifically defined for the i-th node, as a subset of the sampled mesh, that is $U'_i \subseteq M$.

Algorithm 4 VMO-N framework

Main phases: mesh expansion; niching, local optimization and mesh contraction

Further legend: changes over to the original VMO; #.op optional line; permutable lines

Inputs: D-dimensional problem $P \leftarrow (S, \Omega, f)$ as in Section 2, counting the domain bounds B, plus:

$Arg = \{I, T, k, C, A_N, A_L, A_E\}$

I: size of initial mesh
k: number of neighbors
C: stop criteria
T: size of total mesh (set $3.0I \leq T \leq \lceil 3.5I \rceil$ or compute it later)
A_N: set of arguments of the niching method (may be empty)
A_L: set of parameters of the local optimizer (may be empty)
A_A: set of additional arguments (may be empty)

Process:

1. $M \leftarrow form\_initial\_mesh(I)$
2.op $\{E, R\} \leftarrow \{\{\ \}, \{\ \}\}$ // Elitist archive E; second extra list R
3.op $Vrs \leftarrow initialize\_other\_variables()$ // As many as needed
4. **while** no condition $c \in C$ is reached **do**
5.op $M \leftarrow do\_extra(M, \ldots, A_{A1} \subseteq A_A)$ // Further actions on M, e.g. truncation
6. $s^* \leftarrow find\_fittest(M)$
7.op $Gk^* \leftarrow upd\_key\_nodes(M, \ldots, s^*, A_{A2} \subseteq A_A)$ // E.g. nodes 'similar' (in fitness) to s^*
8. $M' \leftarrow M$ // Expanding mesh
9. $\varepsilon \leftarrow distance\_threshold(B, Time_{notion})$
10. **for** $i = 1$ **to** I **do**
11. $Lk_i \leftarrow find\_nearest\_nodes(k, s_i, U_i' \subseteq M)$ // Within universe U_i'
12.~17. $M' \leftarrow local\_expansion(M', s_i, Lk_i)$ // As in Algorithm 1 (lines 8~13)
18. $s_i^* \leftarrow s^*$ **or** $s_i^* \leftarrow select\_key\_node(s_i, Gk^*)$ // Do the second choice iif line 7 is run
19. **if** $s_i \neq s_i^*$ **then**
20. $s_i^G \leftarrow create\_globally(s_i, s_i^*)$ // Global expansion
21. $M' \leftarrow M' \cup \{s_i^G\}$
22. **end if**
23. **end for**
24.op $T \leftarrow figure\_mesh\_size(|M'|, \ldots, A_{A3} \subseteq A_A)$ // If no value is set a priori
25.~27. $M' \leftarrow frontiers\_expansion(M, M', T)$ // As in Algorithm 1 (lines 19~21)
28.op $M' \leftarrow do\_extra(M', Lm, \ldots, A_{A4} \subseteq A_A)$ // Any further process on M'
29. $Nh \leftarrow niche\_formation(M', A_N)$ // Using any desired niching method
30. **for** $j = 1$ **to** $|Nh|$ **do**
31.op $Nh_j \leftarrow local\_optimizer(Nh_j, A_L)$ // On the j-th niche
32.op $Nh_j \leftarrow sort\_by\_fitness(Nh_j)$
33.op $Nh_j \leftarrow adaptive\_clearing(Nh_j, \varepsilon)$
34.op $\{E, R\} \leftarrow upd(E, R, \{Nh_{j,1}, \ldots\}, A_{A5} \subseteq A_A)$ // Update E if master $Nh_{j,1}$ is new. // If lines 34 and 35 permute, $\{Nh_{j,1}, \ldots\}$
35. **end for** // has the masters of all niches
36.op $\{Arg, Vrs\} \leftarrow update(Arg, \ldots, Vrs)$
37. $M \leftarrow form\_next\_mesh(I, \ldots, M', Nh)$ // E.g. randomly, by elitist selection…
38. **if** $|M| < I$ **then**
39. $Ls^{rand} \leftarrow random\_nodes(I - |M|)$
40. $M \leftarrow M \cup Ls^{rand}$ // Complete initial mesh for next cycle
41. **end if**
42.op $\{Arg, Vrs\} \leftarrow update(Arg, \ldots, Vrs)$
43. **end while**

Output:

44. **return** E

The annotated restriction $3.0I \leq T \leq \lceil 3.5I \rceil$ derives from the details of the expansion. Setting $T \leftarrow 3I$ assures a least creation of $2I$ nodes, which is enough to expand locally, globally, and to some extent, from the frontiers. Moreover, $3.0I < T \leq \lceil 3.5I \rceil$ benefits even more the creation of solutions from the frontiers. Actually, $T \leftarrow \lceil 3.5I \rceil$ is more than sufficient to treat both frontiers fully. Instead of setting T as a parameter, now VMO-N can also compute it as needed, for example to precise the exact $\left|Ls^H\right|$ wanted. If $\left|Ls^H\right|$ is very small, $T \lesssim 3.0I$ can be obtained. In addition, even if T is set as a parameter together with I, they may vary along the search (either in line 36 or in line 42).

Other revisions are the operation with several termination criteria for the evolutionary process, and the use of a *while*-loop rather than a *for*-loop to describe it. Although the latter is coherent with our practice of VMO, it has no other impact but gaining descriptive capability for future. The altered framework remains valid to pursue both global and local optima, whereas it allows limiting the search to only one of those types (e.g., by means of lines 34 and 37).

4.2. Hill-Valley-Clustering-Based Variable Mesh Optimization

Algorithm 5 reveals the elementary units of HVcMO, the novel instance of VMO-N, whose competitiveness is later confirmed in Section 6, as to accomplish the third research objective of this study. It is also an evident extension of HillVallEA that integrates some important additions. Thus, HVcMO is seen from the perspectives of its two parents. In this new case of the VMO-N framework:

- two external lists are used to keep the elites and those nodes for rejection, respectively,
- HVC is employed as the niching algorithm,
- the adaptive clearing is not applied,
- a local optimizer is run over each niche, and
- the mesh in the next generation is fully replaced by a new one.

Since the mesh is entirely reset (line 13), an elitist archive is required and the adaptive clearing is needless. The pursuit for diversity recurs, now by the rejection scheme all along the restart of the population, which follows the instructions for HillVallEA, same as the update of the archive. Besides, before niching the expanding mesh, it is enlarged with all the already identified elites (line 39) to use them as attractors while executing the niching method. That is equivalent to evolve a population that consists of two consecutive segments for certain masters of the found niches and for random nodes, respectively, so that the sorting, the truncation and the expansion affect only the latter segment.

Moreover, the search operators of VMO were implemented the same as for [20–22]. The corresponding formulations given in [20] largely apply, except for some changes that involve mainly the local expansion. Among them, the distance threshold is based on [21,22], as:

$$\varepsilon \leftarrow \frac{1}{D} \sum_{j=1}^{D} \varepsilon_j \tag{10}$$

where D is the poblem dimension and each component ε_j denotes a portion of the amplitude of the domain of the j-th variable whose upper and lower bounds are respectively b_j and a_j. That fraction depends on the current count of function evaluations, out of a fixed maximum number (*MaxFEs*):

$$\varepsilon_j \leftarrow \begin{cases} \left(b_j - a_j\right)/2, & FE_{used} < 0.15MaxFEs \\ \left(b_j - a_j\right)/4, & 0.15MaxFEs \leq FE_{used} < 0.30MaxFEs \\ \left(b_j - a_j\right)/8, & 0.30MaxFEs \leq FE_{used} < 0.60MaxFEs \\ \left(b_j - a_j\right)/16, & 0.60MaxFEs \leq FE_{used} < 0.80MaxFEs \\ \left(b_j - a_j\right)/100, & 0.80MaxFEs \leq FE_{used} \end{cases} \tag{11}$$

Algorithm 5 HVcMO

Main phases: ■ mesh expansion ■ niching, local optimization and mesh contraction
Further legend: ■ main extensions to HillVallEA

Inputs: D-dimensional problem $P \leftarrow (S, \Omega, f)$ as in Section 2, including the domain bounds B, plus:
$\mathcal{A}$: local optimizer
$C = \{MaxFE\}$ (budget as maximum function evaluations)
$\{I, k\}$ as in Algorithm 1
H_{amp}: amplitude rate of the frontiers together ($H_{amp} \leq 0.5$)
$\tau_c^{\mathcal{A}}$: selection fraction for $\mathcal{A}$
H_{max}: maximum number of frontier nodes to expand
$\mathcal{N}_c^{\mathcal{A}}$: cluster size for $\mathcal{A}$
$\mathcal{N}_c^{ini}$: initial fraction of the population size of $\mathcal{A}$
$\mathcal{N}_c^{inc}$: cluster size increment factor
$\mathcal{N}^{inc}$: mesh size increment factor
Tol: tolerance level

Process:

1. $\mathcal{N}_c \leftarrow \mathcal{N}_c^{ini} * \mathcal{N}_c^{\mathcal{A}}$ // Actual population size for $\mathcal{A}$
2. $\tau \leftarrow \tau_c^{\mathcal{A}}$
3. $E \leftarrow \{\ \}$ // Elitist archive
4. $R \leftarrow \{\ \}$ // Set of nodes used for rejection
5. $min\_I \leftarrow I$
6. **while** $FE_{used} \leq MaxFE$ **do** // If remaining budget is thought
7. **if** $FE_{toExpand}(I) > MaxFE - FE_{used}$ **then** // too little to expand some of the
8. $I \leftarrow shrink\_mesh\_size(I, MaxFE, FE_{used})$ // new nodes, properly reduce I
9. **if** $I < min\_I$ **then**
10. **break while** // Stop if the least size is passed
11. **end if**
12. **end if**
13. $M \leftarrow sample\_uniform\_reject(I, D, R)$
14. $R \leftarrow M$ // Reset the rejection set
15. $descending\_sort\_by\_fitness(M)$
16. $M \leftarrow truncate(M, \tau)$ // Select the $\lceil I * \tau \rceil$ nodes
17. $s^* \leftarrow find\_fittest(M)$
18. $|Ls^H| \leftarrow min(\lceil |M| * H_{amp} \rceil, H_{max})$
19. ~ 38. $M' \leftarrow expand(M, B, k, s^*, |Ls^H|, to\_sl11, sl24)$ where $sl24$ is "$T \leftarrow |M'| + |Ls^H|$" and to_sl11 means "$U_i' \leftarrow truncate(M, \tau)$" // Expand M as in Algorithm 4 // (lines 8~27), running its line 24 as // $sl24$, and using to_sl11 to set // U_i' in its line 11
39. $M' \leftarrow M' \cup E$
40. $Nh \leftarrow HillValleyClustering(M')$
41. **for** $j = 1$ **to** $|Nh|$ **do**
42. **if** $Nh_{j,1} \notin E$ **then** // If $Nh_{j,1}$ is recent, run $\mathcal{A}$ to
43. $cand \leftarrow \mathcal{A}(Nh_j, \mathcal{N}_c, Tol)$ // enhance Nh_j, and take its best
44. $Cnd \leftarrow Cnd \cup \{cand\}$ // node as a candidate optimum
45. **end if**
46. **end for**
47. $\mathcal{N}_u \leftarrow update\_elitist\_archive(E, Cnd, Tol)$ // Count elites replaced or added
48. $e^* \leftarrow get\_fittest(E)$
49. $R \leftarrow R \cup \{e^*\} \cup \{e \mid just\_added(e, E), e \neq e^*\}$ // Update the rejection set
50. **if** $\mathcal{N}_u = 0$ **and** *mesh size never decreased* **then**
51. $I \leftarrow I * \mathcal{N}^{inc}$ // If E is same, and I has not
52. $\mathcal{N}_c \leftarrow \mathcal{N}_c * \mathcal{N}_c^{inc}$ // reduced, rise population sizes
53. **end if**
54. **end while**

Output:

55. **return** E

Besides, for every node s_i to be expanded locally by the *F*-operator, the list of neighbors Lk_i is not determined from the entire mesh but from a certain universe U'_i. In HVcMO, regardless of the node at hand, such a universe is the fittest τ-fraction of the mesh. As for that, we use the selection fraction defined for the truncation of the initial mesh, but they may also be set unequally. Not all the solutions in U'_i have global importance, even for relatively short values of τ. For that reason, that universe should not be confused with a set of key nodes. The larger τ is, the more nodes with no global relevance are included in U'_i. The choice of that quasi-local *F*-operator is mostly moved by our concern about the running time. Future works should evaluate the effects of other ways to decide U'_i.

The formulations for the *G*-operator and the *H*-operator remains unaltered, but the size of frontiers affected by the latter is jointly delimited by their fixed amplitude ratio H_{amp}. In the previous studies about VMO, the value of T is fixed as a parameter, informing the possible largest length of the mesh after the expansion. Thus, the extent of the creation of nodes from the frontiers, i.e., $\left|Ls^H\right|$, is basically figured as the difference of T and the size of the enlarging mesh (Algorithm 1, line 19). In that case, the proportion of the mesh taken as frontiers is influenced by T. Conversely, in HVcMO, $\left|Ls^H\right|$ is computed as the lowest value between the percentage of the initial mesh indicated by H_{amp}, and a firm upper limit (H_{max}) for the size of the frontiers (Algorithm 5, line 18). Next, in lines 19~38 that are equivalent to lines 8~27 in Algorithm 4, it is marked how to calculate T. Note that it is just a formalism to trace T from Algorithm 5 to Algorithm 4, and then from Algorithm 4 to Algorithm 1, as to figure $\left|Ls^H\right|$. In practice, after calculating $\left|Ls^H\right|$, HVcMO does not utilize any T at all.

From the viewpoint of HillVallEA, the biggest augmentation in HVcMO is the effecting of the search operators of VMO (lines 18~39) over the truncated population. As of the population is restarted randomly, HillVallEA uses a couple of important strategies, i.e., the rejection mechanism to sparse the solutions through unexplored areas of the search space, and the subsequent truncation to process only the best part of the population. The adding of the expansion operators of VMO as another preparation step before niching intends to improve the quality of the population as well. However, it provokes that the number of nodes increases, and then also the consumption of the budget and the execution time. That consequence applies for any specific sample size at a certain moment, but is not necessarily valid for the entire evolutionary optimization. Our conjectures about it are delineated as:

- *Hypothesis 1:* Using additional search operators to enhance the population before niching may reduce the total number of solutions throughout the optimization process by HillVallEA.
- *Hypothesis 2:* If the reduction of the overall number of solutions is big enough, the total execution time of HillVallEA should also decrease, while keeping quite similar multimodal capability.

To evaluate them (in Section 6), we borrowed search procedures from VMO. Others may be used instead, e.g., crossover mechanisms designed for GAs. Thus, a research avenue for HillVallEA begins. Another variation by HVcMO concerns the diminution of the mesh size when the remaining budget is insufficient to deal with a new population having the current length. This choice was indeed contemplated by Maree et al. in [48], but discarded, as they considered fruitless to sample such small populations at the end of the optimization process. In case of HVcMO, the size of the mesh properly decreases (line 8) when the available budget is insufficient not only to sample a population with the current size, but also to expand at least part of the new nodes, via VMO. Even without any deep analysis about it, we implemented that modification based on some empirically observed benefits. Besides, it slightly increments the overall number of solutions and thus, the running time. Therefore, that means a practical opportunity to prove that even with a forced longer execution time (beyond that provoked by VMO itself), HVcMO can run faster than HillVallEA. However, future works should verify the actual advantages of keeping such a late shrinkage of the population in HVcMO.

As an aftermath, the condition to increase the size parameters (line 50) is altered. In HillVallEA, that occurs if no new peak is detected. In HVcMO, it happens if also the size of the population was never reduced, since it is senseless to push a larger mesh again after it was previously shortened due to a lack of budget. On the other hand, one more extension pretends to reinforce the potential of the

rejection while sampling the population. Not only the solutions sampled in the preceding iteration are considered, but also each master representing a newly discovered peak (in the current iteration), together with the best elite ever found (line 49). A closing comment clarifies about HVcMO20a, which is nothing else but the HVcMO algorithm where AMaLGaM-Univariate performs as the local optimizer and the parametric specifications for HillVallEA19 are widely adopted, as detailed below.

5. Experimental Setup

The general elements of the experimental analysis are explained in this section. They include the setting of parameters for HVcMO20a, the suite of benchmark problems, the baseline methods, the performance criteria, and the statistical tests for comparisons. This work follows the procedures of the Competition on Niching Methods for Multimodal Optimization within GECCO [28], which extensively embraces the instructions in [40]. For any single run, the tried algorithm stops when a given budget is finished. Stated as a certain maximum number of functions evaluations (*MaxFE*), the budget for every test problem is specified in Table 2. Every algorithm is run 50 times over each test problem. The outputs are assessed at five levels of accuracy: 1.0×10^{-1}, 1.0×10^{-2}, 1.0×10^{-3}, 1.0×10^{-4} and 1.0×10^{-5}, and the results are averaged over a given number of runs ($NR \leftarrow 50$) at every level of accuracy. Such a concept of accuracy is meant only to evaluate the outputs of any multimodal optimizer. However, methods that use certain thresholds, e.g., the tolerance in HillVallEA, may set them by considering those values of accuracy as a reference.

Table 2. Characteristics of the test problems and budget allowance. Each value in red represents a search space volume that is either small or large; if blue, it denotes a volume of a medium size.

Id	f	Name	D	Domain Bounds	*Vol*	#GOpt	#LOpt	MaxFE
1	$F_1(x)$	Five-Uneven-Peak Trap	1	$x \in [0, 30]$	30.00	2	3	5.0×10^4
2	$F_2(x)$	Equal Maxima	1	$x \in [0, 1]$	1.00	5	0	5.0×10^4
3	$F_3(x)$	Uneven Decreasing Maxima	1	$x \in [0, 1]$	1.00	1	4	5.0×10^4
4	$F_4(x, y)$	Himmelblau	2	$x, y \in [-6, 6]$	144.00	4	0	5.0×10^4
5	$F_5(x, y)$	Six-Hump Camel Back	2	$x \in$ [-1.9, 1.9]; $y \in [-1.1, 1.1]$	8.36	2	5	5.0×10^4
6	$F_6(\vec{x})$	Shubert	2	$x_1, x_2 \in [-10, 10]$	4.0×10^2	18	many	2.0×10^5
7	$F_7(\vec{x})$	Vincent	2	$x_1, x_2 \in [0.25, 10]$	95.06	36	0	2.0×10^5
8	$F_6(\vec{x})$	Shubert	3	$x_1, x_2, x_3 \in [-10, 10]$	8.0×10^3	81	many	4.0×10^5
9	$F_7(\vec{x})$	Vincent	3	$x_1, x_2, x_3 \in [0.25, 10]$	926.86	216	0	4.0×10^5
10	$F_8(\vec{x})$	Modified Rastrigin	2	$x_1, x_2 \in [0, 1]$	1.00	12	0	2.0×10^5
11	$F_9(\vec{x})$	Composition Function 1	2	$x_1, x_2 \in [-5, 5]$	1.0×10^2	6	many	2.0×10^5
12	$F_{10}(\vec{x})$	Composition Function 2	2	$x_1, x_2 \in [-5, 5]$	1.0×10^2	8	many	2.0×10^5
13	$F_{11}(\vec{x})$	Composition Function 3	2	$x_1, x_2 \in [-5, 5]$	1.0×10^2	6	many	2.0×10^5
14	$F_{11}(\vec{x})$	Composition Function 3	3	$x_1, x_2, x_3 \in [-5, 5]$	1.0×10^3	6	many	4.0×10^5
15	$F_{12}(\vec{x})$	Composition Function 4	3	$x_1, x_2, x_3 \in$ [-5, 5]	1.0×10^3	8	many	4.0×10^5
16	$F_{11}(\vec{x})$	Composition Function 3	5	$x_1, x_2, \ldots, x_5 \in$ [-5, 5]	1.0×10^5	6	many	4.0×10^5
17	$F_{12}(\vec{x})$	Composition Function 4	5	$x_1, x_2, \ldots, x_5 \in$ [-5, 5]	1.0×10^5	8	many	4.0×10^5
18	$F_{11}(\vec{x})$	Composition Function 3	10	$x_1, x_2, \ldots, x_{10} \in$ [-5, 5]	1.0×10^{10}	6	many	4.0×10^5
19	$F_{12}(\vec{x})$	Composition Function 4	10	$x_1, x_2, \ldots, x_{10} \in$ [-5, 5]	1.0×10^{10}	8	many	4.0×10^5
20	$F_{12}(\vec{x})$	Composition Function 4	20	$x_1, x_2, \ldots, x_{20} \in$ [-5, 5]	1.0×10^{20}	8	many	4.0×10^5

5.1. Configuration of Parameters of HVcMO20a

Equivalent to N in HillVallEA, the size of the initial mesh is set as $I \leftarrow 64$, setting a selection fraction of $\tau_c^{\mathcal{A}} \leftarrow 0.35$ The number of neighbors of each node when affected by the local expansion is $k \leftarrow 3$, the same as in [20–23]. HVcMO20a takes AMaLGaM-Univariate as the local optimizer and then, the size of the population to enhance every niche (by means of HVC) is $N_c^{\mathcal{A}} \leftarrow 10\sqrt{D}$ (the values of D are shown in Table 2), and its initial fraction is $N_c^{ini} \leftarrow 0.8$. The increment factors for the length of the overall population and the cluster size are fixed as $N^{inc} \leftarrow 2.0$ and $N_c^{inc} \leftarrow 1.1$, respectively.

The level of tolerance is set to 1.0×10^{-5}, except for the later estimate of the time ratio (see Section 6.3). The domain bounds (B) for the variables of each problem are also given in Table 2.

Furthermore, the frontiers are jointly delimited by an amplitude rate of $H_{amp} \leftarrow 0.05$, and the nodes to generate from them are at most $H_{max} \leftarrow 50$. Hence, while in the evolutionary optimization the length of the initial mesh grows in the range $64 \leq I \leq 2857$, it is truncated to $23 \leq |M| \leq 1000$ nodes and therefore, the number of nodes to expand from the frontiers is $2 \leq \left|Ls^H\right| \leq 50$. For larger initial meshes, $\left|Ls^H\right|$ keeps at constant value of 50. This choice of effecting the H-operator over a few solutions responds to the outcomes by previous works and some analytical observations. Despite the frontier operator was found useful for global optimization with VMO [32,51], there is no evidence of its impact on the VMO-based multimodal optimizers. Indeed, that represents another pending research issue, in particular since we noticed a detriment of the performance of HVcMO in some cases when the H-operator is applied on the large scale. We keep the usage of the frontier operator at a low rate, based on such yet incomplete findings, and also on the outcomes unveiled in [51], where it is proved the less influential among the search operators of VMO for global optimization.

5.2. Benchmark Multimodal Problems

A standardized test suite is used, being $NF \leftarrow 20$ the number of problems. Table 2 summarizes their main features, detailed in [40], plus the *MaxFE* specified as the budget afforded. The objective functions are formulated in [40], as well. We add information about the volume of the box-bounded search space; it is expressed as the product of the amplitudes of the domains of all variables:

$$Vol \leftarrow \prod_{j=1}^{D} \left(b_j - a_j\right) \tag{12}$$

being D the dimension, and b_j and a_j the respective upper and lower bounds of the j-th variable. Values are highlighted (in red or blue) according to the volume, empirically seen as small ($Vol \leq 90$), medium ($90 < Vol \leq 10^3$) or large ($10^3 < Vol \leq 10^{20}$). This issue gains importance in Section 6.

5.3. Baseline Methods

As part of the experimental analysis, HVcMO20a is compared to the other VMO-based multimodal optimizers, namely NC-VMO and VMO-ASD (the previous instances of the VMO-N framework), and also the NVMO method. Besides, HVcMO20a is contrasted with the HillVallEA19 algorithm and its predecessor, HillVallEA18, as well as with other remarkable metaheuristics such as NEA2+, proposed by M. Preuss in [52], and RS-CMSA, introduced by Ahrari et al. in [53]. The other included baselines are SDE-Ga and ANBNWI-DE, by Kushida, respectively ranked first and second in the last two editions of the abovementioned competition, whose results are reported in [54].

5.4. Major Performance Criteria

The following description concerns the standard measures utilized for comparing HVcMO20a with the group of baseline optimizers, while additional criteria are later used to contrast it only with HillVallEA19, regarding their running times (Section 6.3). According to the most recent procedures indicated for the referred competition, every method is assessed by taking into account three scenarios defined by these scores: the peak ratio (PR), the (static) $F1$ measure and the dynamic $F1$. They are bounded in $[0,\ 1]$; the larger, the better. Among them, the PR allows contrasting new algorithms with a larger set of earlier multimodal optimizers, even if not recent ones, since the other two criteria were just adopted in the last years. For any specific run, the PR is the percentage of the number of peaks found (NPF) out of the number of known peaks (NKP), keeping in mind that such peaks represent global optima only:

$$PR \leftarrow \frac{NPF}{NKP} \tag{13}$$

Additionally, the well-known $F1$ statistic measure is re-formulated in this context as follows:

$$F1 \leftarrow \frac{2 * PR * SR'}{PR * SR'} \tag{14}$$

where given the set of output solutions (OS), i.e., the presumed global optima returned by the algorithm, the success rate SR' of OS tells the fraction of actual global optima found with respect to the count of output solutions:

$$SR' \leftarrow \frac{NPF}{|OS|} \tag{15}$$

It is important to distinguish SR' from the success rate (SR) of runs [40], a formerly used measure. Furthermore, after a run is finished, the achieved OS set is entirely used to compute the static $F1$, while the dynamic $F1$ is progressively calculated at the moments when solutions are discovered. Thus, such a calculation considers the count of function evaluations (FE_i) at the instant i, with $1 \leq i \leq |OS|$, which corresponds to the i-th found optima in OS. The set $OS_{[1:i]}$ then consists of the first i optima located. Finally, the dynamic $F1$ is figured as the area under the curve divided by the $MaxFE$ allotted:

$$dynF1 \leftarrow \frac{\left(MaxFE - FE_{|OS|}\right)F1(OS) + \sum_{i=2}^{|OS|}(FE_i - FE_{i-1})F1\left(OS_{[1:i-1]}\right)}{MaxFE} \tag{16}$$

5.5. Statistical Validation

Regardless of the measures, the advices by Demšar [55] for comparing classifiers using non-parametric tests turned into a universal practice to assess methods in several fields, e.g., evolutionary computation [56]. However, as we alerted [22], such statistical comparisons are common in studies on global optimization, but sporadic in those about multimodal optimizers. For the validation of the outcomes by HVcMO20a, we apply the Wilcoxon signed-ranks test to contrast pairs of algorithms, and the Friedman test to detect significant differences between a group of methods. The non-parametric analyses aim to reject the null-hypothesis (H_0) that the compared algorithms perform similarly. If the Friedman test does it, we use the post-hoc Nemenyi test to identify which methods differ significantly, concerning the measure at hand. The election of this last procedure responds also to the intention of contrasting, in unison, the earlier versions of HillVallEA with other multimodal optimizers, from a statistical perspective. Here, we set the significance level at 5%, i.e., the confidence level is $\alpha \leftarrow 0.05$.

6. Discussion of Results

The output files of HillVallEA19 that led to the statements presented in [27], are now available online [54]. However, it was necessary to compute further rough data, e.g., the population size at the end of each iteration of the algorithm, as to conduct the analysis given in Section 6.3. For that reason, we executed HillVallEA19 again by using its original program [48]. Hence, for a matter of homogeneity, we calculated the PR, $F1$ and dynamic $F1$ for the new output data in order to use their values also in the Section 6.2, instead of the ones published in [27]. As expected, the values for those metrics in both studies are quite similar. In view of that, if the analysis related to such measures is replicated, assuming the numbers in [27], it will lead to equivalent ends. As Supplementary Materials, the output files of our runs of HillVallEA19 and HVcMO20a are accessible online [57], in the required format [28], where the times reported have no other use than checking the correct order of the output solutions recorded; those values of time are not considered for the experiments in this study.

6.1. Outperforming NVMO and Earlier Instances of VMO-N

According to the availability of performance data, HVcMO20a is contrasted with the previous VMO-based multimodal optimizers by means of the peak ratio only. Table 3 shows the average assessments (at all accuracy levels) of all runs over the whole test suite, for each algorithm. In case

of NC-VMO and VMO-ASD, the comparisons are based only on the results for 6 and 8 problems, respectively, which are reported in [22]. On the other hand, the results considered for NVMO involve the full validation suite of 20 problems, as published in [23]. The results by HVcMO20a over every single benchmark function are detailed in Table A1 from Appendix A.

Table 3. Peak ratios reached by the VMO-based multimodal optimizers, averaged at all accuracy levels over sets of 6, 8 and 20 benchmark problems.

{Problem Ids}	{1, 2, 3, 4, 5, 10}		{1, 2, 3, 4, 5, 7, 9, 10}		{1 ~ 20}	
	NC-VMO	**HVcMO20a**	**VMO-ASD**	**HVcMO20a**	**NVMO**	**HVcMO20a**
Mean PR	0.696	1.000	0.523	0.994	0.698	0.885

Although the overall PR values put HVcMO20a on top of all these methods, the application of the Wilcoxon test over each pair of algorithms is required to deepen the analysis. Table 4 confirms that HVcMO20a significantly outperforms both VMO-ASD and NVMO, regarding the peak ratio. About the remaining evaluation (HVcMO20a vs. NC-VMO), it takes into account six problems; for two of them, those methods achieve equal outputs. Thus, there are only four relevant comparisons, a sample size that is insufficient to make a reliable computation of the Wilcoxon test. Alternatively, it is possible to calculate it with the 6 comparisons as relevant by splitting the ranks of such two ties evenly among the sums R+ and R−. In that variant, the R+ would be 19.5, while both R− and the W statistic would be equal to 1.5, which is greater than 0, the critical value of the Wilcoxon test for 6 paired comparisons at $\alpha \leftarrow 0.05$, making the test unable to reject the null-hypothesis. Hence, no significance difference between the peak ratios scored by HVcMO20a and by NC-VMO are detected with the studied data.

Table 4. Wilcoxon test regarding PR. The sum of ranks of comparisons where HVcMO20a outdoes the other algorithm (either VMO-ASD or NVMO) is R+, and R− is the opposite sum. If the W statistic is equal or less than the corresponding critical value, the null-hypothesis is rejected.

HVcMO20a vs. VMO-ASD					**HVcMO20a vs. NVMO**				
R+	**R−**	**W**	**Critical Value**	**H_0**	**R+**	**R−**	**W**	**Critical Value**	**H_0**
21	0	0	0	Rejected	118	2	2	25	Rejected

6.2. Further Baseline Comparison

Different from above, HVcMO20a is contrasted with the previous variants of HillVallEA and other outstanding meta-heuristics by means of the PR and the two other scenarios, i.e., the static and the dynamic $F1$. The mean results of our executions of HVcMO20a, HillVallEA19 and ANBNWI-DE over the benchmark problems are shown in Table A1, in Appendix A. For ANBNWI-DE, we computed the performance scores by using the output data available in [54], same as for the other baseline methods: NEA2+, RS-CMSA and SDE-Ga. However, we do not report the calculations for the last three algorithms because they match those published in [27], in relation to them.

Table 5 exhibits the average measurements for the multimodal optimizers, considering all accuracy levels and the whole set of problems. The group of HillVallEA methods, including HVcMO20a, beat the rest of the algorithms in every scenario. Among them, HillVallEA19 shows the best performance, closely followed by HVcMO20a and HillVallEA18, in that order. However, is the advantage of the HillVallEA family over the remaining algorithms significant? How about the differences within the three variants of HillVallEA?

Table 5. Scores averaged at all accuracy levels over the entire validation suite.

Mean	NEA2+	RS-CMSA	SDE-Ga	ANBNWI-DE	HillVallEA18	HillVallEA19	HVcMO20a
PR	0.807	0.856	0.833	0.855	*0.885*	0.890	*0.885*
F1	0.855	0.911	0.884	0.887	*0.930*	0.933	*0.930*
dynF1	0.806	0.829	0.376	0.727	0.869	0.883	*0.876*

The Friedman test confirms the existence of significant differences in the pool of methods at every scenario (see Table 6). Therefore, a post-hoc examination should determine which algorithms perform significantly distinct. That is clarified in Figures 4–6, the graphical representations of the implication of the Nemenyi test for the scenarios of *PR*, *F*1 and dynamic *F*1, respectively.

Table 6. Friedman test. Since p-value < 0.05 (α), each null-hypothesis is rejected.

Scenario	p-Value	H_0
PR	3.674×10^{-6}	Rejected
F1	1.907×10^{-7}	Rejected
dynF1	4.749×10^{-11}	Rejected

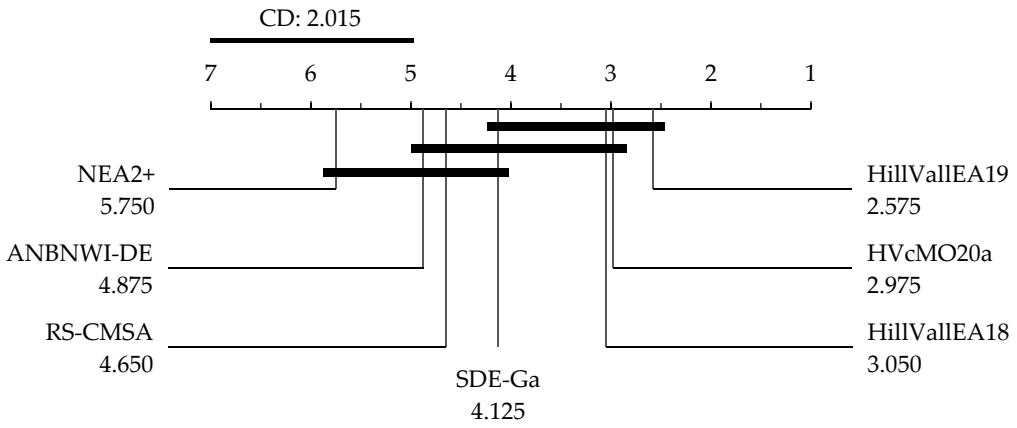

Figure 4. Nemenyi test regarding PR. Connected optimizers are not statistically different.

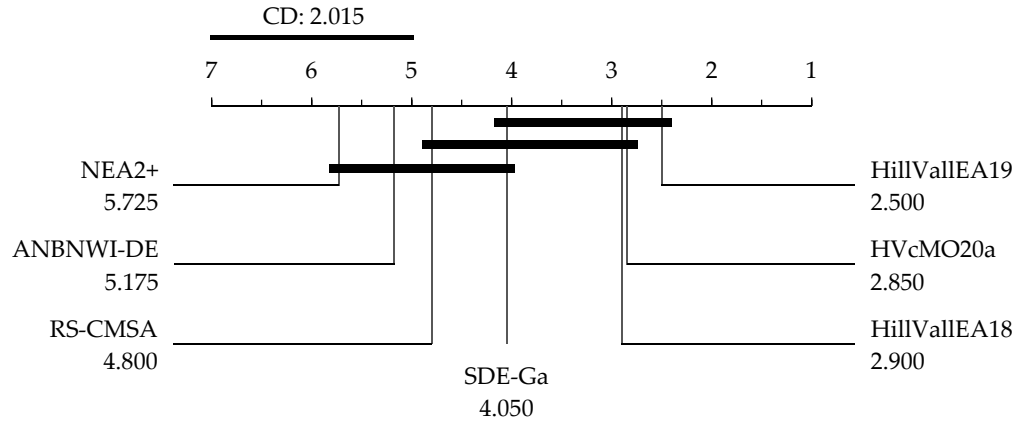

Figure 5. Nemenyi test regarding F1. Connected optimizers are not statistically different.

For seven algorithms and 20 comparisons (benchmark problems), the critical difference (CD) for the Nemenyi procedure at $\alpha \leftarrow 0.05$ is 2.015. Any two algorithms perform significantly distinct if the distance between their average ranks is at least CD. Although HillVallEA19, HVcMO20a and HillVallEA18 confirm to rank before all the other methods, there is no significant difference between

the three of them at any scenario. Thus, the performances of such three algorithms are statistically equivalent. Regarding the peak ratio, HillVallEA19 significantly outdoes RS-CMSA, ANBNWI-DE and NEA2+. Both HVcMO20a and HillVallEA18 are significantly better than NEA2+.

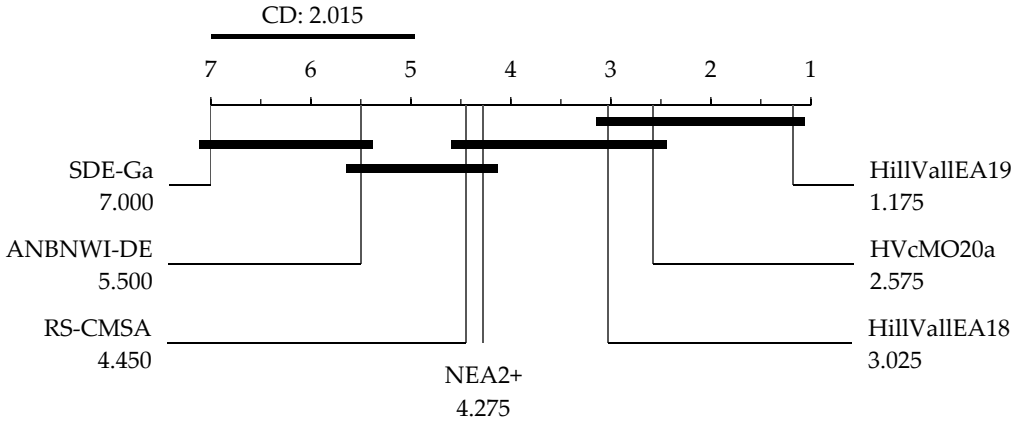

Figure 6. Nemenyi test regarding dynamic F1. Connected optimizers are not statistically different.

With respect to the *F*1 measure, HillVallEA19 statistically beats again RS-CMSA, ANBNWI-DE and NEA2+. The outcomes by the HVcMO20a and the HillVallEA18 algorithms are statistically better than those achieved by ANBNWI-DE and by NEA2+. Moreover, in the scenario of the dynamic *F*1, HillVallEA19 is significantly better than NEA2+, RS-CMSA, ANBNWI-DE and SDE-Ga, while the superiorities of HVcMO20a and HillVallEA18 over both ANBNWI-DE and SDE-Ga are significant.

Besides being the best VMO-based multimodal optimizer, these results confirm that HVcMO20a beats several prominent algorithms from the state-of-the-art, significantly in some cases. In so doing, the third research objective (the pursue of a competitive variant of VMO-N) is accomplished.

The application of statistical comparisons keeps as a research debt in the arena of multimodal optimization. The utility of the non-parametric procedures goes beyond the discovery of significant contrasts between the performances of the methods. For instance, giving the mean scores in Table 5, HVcMO20a and HillVallEA18 are equal in terms of *PR* and *F*1, while ANBNWI-DE is better than SDE-Ga in such scenarios. However, Figures 4 and 5 clarify that indeed HVcMO20a is rated before HillVallEA18, and SDE-Ga is better placed than ANBNWI-DE, in view of the individual ranks of their values for every test problem, instead of the average achievements.

6.3. HVcMO20a or HillVallEA19? When to Apply Each?

It is evident now the gain of putting the formulations of HillVallEA on the VMO-N framework, resulting in the effective HVcMO20a algorithm. The benefit of using the search operators of VMO in HillVallEA is not yet clear though. Summarily, that contribution comes in terms of the execution time.

6.3.1. When Counting Function Evaluations Is Not Enough

The convergence speed of an optimizer is a common way to get an idea of its rapidness. However, making a suitable formulation of that measure is more difficult in multimodal optimization than in global optimization. In [40], it is defined as the average number of function evaluations (*FE*) needed to locate all global optima. If the optimizer cannot find all the desired optima, *MaxFE* (the budget) is assumed as the *FE* for that run. However, there are various situations in which it gives a wrong notion, e.g., it considers the count of optima only if all of them are located. For example, if any pair of multimodal optimizers reach the fixed budget (and stop), after respectively finding the 30% and the 60% of the wanted optima, the convergence speed for both of them is interpreted the same (*MaxFE*). However, one of them seems able to converge (to all optima) first, if the budget were larger. Yet hard, an alternative for this is to substantially increase the budget for a better convergence analysis, like in [24].

In addition, what does one function evaluation represents in terms of time? Is it possible that a certain optimizer burns the same budget than another, but converges more rapidly to the same optima?

Table 7 shows the mean numbers of function evaluations by both HillVallEA19 and HVcMO20a over the entire test suite, and over problems grouped by the search space volume. HillVallEA19 uses less budget than HVcMO20a, but the average count of iterations when HillVallEA19 runs is larger. Besides, HillVallEA19 fails (no new elite is found) in more iterations than HVcMO20a. Thus, HillVallEA19 increases the population more often and what is more relevant, to a larger extent. At the start, they both use 64 individuals; if the current iteration fails, they become 128, then 256, and so on. Adding 703 solutions (in average) to the population in HillVallEA19 whenever it fails, but only 497 in HVcMO20a, indicates that smaller unsuccessful populations are doubled in HVcMO20a, i.e., at earlier moments. How many solutions do they handle in total? How much does it delay them?

Table 7. Mean values of budget usage, failed iterations and population increase after failure.

Volume of Search Space	Function Evaluations		Iterations (Failed Iterations)		Population Increase	
	HillVallEA19	HVcMO20a	HillVallEA19	HVcMO20a	HillVallEA19	HVcMO20a
medium	114,080	116,480	19.11 (4.49)	17.32 (3.70)	684	401
small + large	51,805	53,751	6.84 (3.58)	6.63 (3.38)	723	593
all sizes	82,942	85,115	12.98 (4.03)	11.98 (3.54)	703	497

The convergence speed cannot answer such interrogations. The alternative to follow depends on the situation. In case of HillVallEA19 and HVcMO20a, it is possible to move forward by analyzing the time complexity, that can be preliminary understood as $O\left(n^2\right)$, where n represents N for HillVallEA19 and I for HVcMO20a. Beyond such a brief statement, comparing the execution times of such algorithms requires to study exhaustively their time functions, which have to be carefully defined. Since both functions are in the order of n^2, ascertaining their dominant coefficients is the key to compare the algorithms with respect to the running time. Otherwise, that can be done through a vast empirical analysis of the times consumed by the programs that implement HVcMO20a and HillVallEA19, which is the way followed in this paper. Regardless of the choice taken, the primary aim is to estimate the ratio of $t_{HVcMO20a}(n)$ to $t_{HillVallEA19}(n)$, which respectively indicates the execution time by HVcMO20a, and the running time by HillVallEA19. Formally, the time ratio is:

$$t_{ratio} \leftarrow \frac{t_{HVcMO20a}(n)}{t_{HillVallEA19}(n)} \tag{17}$$

From the empirical viewpoint, it is needed a vast number of paired time comparisons that should also be diverse, e.g., involving several test functions and varying parameters (tolerance, populations size, etc.). The more contrasting cases are considered, the more reliable the estimation of the ratio is. Since a full harmonization is impossible, a range of actions should be done to track and to examine sufficient rough running times of the programs in a fair manner. In our analysis, that is intended by:

- executing single iterations of them, instead of the whole evolutionary process,
- conducting the experiment over populations of certain sizes, to be exact $n \in \left\{2^6, 2^7, 2^8, 2^9, 2^{10}\right\}$,
- generating 50 distinct populations for every single sample size,
- running the programs over each of the 20 benchmark problems using the same populations,
- replicating the process for five levels of tolerance set equal to the accuracy levels, for a total of 25,000 runs per program (the tolerance influences AMaLGaM-Univariate and thus, the overall process),
- effecting the 50,000 runs in turn and on the same computer, i.e., using the same specifications of hardware and software,
- operating no other computational process, apart from those controlled by the system,

- reusing much of the source code of HillVallEA19 to program the common aspects in HVcMO20a, to reduce the influence that the skills of the programmer have over the execution time, and by
- estimating the ratio based not only on mean calculations but also on every single paired contrast,
- excluding the outliers during the examination of the resultant measurements.

6.3.2. The Time Ratio

The first attempt to decide the proportion of the time by HVcMO20a with respect to the time by HillVallEA19 concerns the ratio of means (*RoM*) and the mean of ratios (*MoR*) metrics. In this study, they can be defined as follows, in a wide manner:

$$RoM \leftarrow \frac{\sum_{exp=lowerExp}^{upperExp} \sum_{pId=lowerId}^{upperId} \sum_{run=1}^{NR} t_{HVcMO20a}(2^{exp})_{pId,run}}{\sum_{exp=lowerExp}^{upperExp} \sum_{pId=lowerId}^{upperId} \sum_{run=1}^{NR} t_{HillVallEA19}(2^{exp})_{pId,run}} \tag{18}$$

$$MoR \leftarrow \frac{\sum_{exp=lowerExp}^{upperExp} \sum_{pId=lowerId}^{upperId} \sum_{run=1}^{NR} \frac{t_{HVcMO20a}(2^{exp})_{pId,run}}{t_{HillVallEA19}(2^{exp})_{pId,run}}}{(upperExp - lowerExp + 1) * (upperId - lowerId + 1) * NR} \tag{19}$$

where $lowerExp, upperExp \in \{6, 7, 8, 9, 10\}$, with $upperExp \geq lowerExp$. Thus, it can be figured with respect to only one specific population size, or regarding several (all) sizes. Besides, *pId* denotes the test problem, with $lowerId, upperId \in \{1, 2, \cdots, 19, 20\}$ and $upperId \geq lowerId$, so that the times for either only one benchmark problem or the overall test suite can be taken. Finally, *NR* represents the number of runs for which the checked running times are considered.

Table A2 (Appendix B) reveals the values of *MoR* and *RoM* for each test problem, seeing the population sizes by separate, and all together. They consider the times for all the runs at every level of tolerance ($NR \leftarrow 250$), and also for those runs of the programs adopting a tolerance of 1.0×10^{-5} ($NR \leftarrow 50$). The observations for 50 and for 250 runs coincide, evidencing the reliability of the experiment. Such a match suggests that in terms of time both programs respond stably in the same way to the variation of the tolerance, as long as the processed populations and the parametric setups (including the adjusted tolerance) are the same. The highest values of *MoR* and *RoM* for every population size are reported in Table 8, together with their overall rates, i.e., considering all the problems. We skip the least values as they might represent outliers. Preliminary, $1.30 < t_{ratio} \leq 1.65$, but since these are mean rates, deciding the ratio in that range might lead to an undesired favoritism for HVcMO20.

Table 8. General values of ratio of means (RoM) and mean of ratios (RoM).

	Pop Size: 2^6		**Pop Size: 2^7**		**Pop Size: 2^8**		**Pop Size: 2^9**		**Pop Size: 2^10**		**All Pop Sizes**	
	RoM	**MoR**	**RoM**	**MoR**	**RoM**	**MoR**	**RoM**	**MoR**	**RoM**	**MoR**	**RoM**	**MoR**
Considering 50 Runs per Function for Every Population Size, Using a Tolerance of 1.0×10^{-5}												
Max	1.616	1.635	1.536	1.562	1.488	1.489	1.486	1.487	1.388	1.391	1.412	1.468
Overall	1.091	1.299	1.088	1.324	1.188	1.328	1.291	1.353	1.301	1.319	1.268	1.325
Considering 250 Runs per Function: 50 Runs for Each Population Size, Using the 5 Levels of Tolerance												
Max	1.609	1.626	1.498	1.523	1.495	1.503	1.464	1.465	1.407	1.409	1.424	1.457
Overall	1.065	1.305	1.297	1.314	1.345	1.352	1.334	1.337	1.314	1.335	1.273	1.329

For every single pair of runs, we find the interval its ratio belongs to, among nine possible ranges whose amplitudes were decided in the experimental phase (see Table A3, Appendix B). It shows that over the 60.80% of the ratios are in the range (1.30, 1.65], confirming the previous analysis of the mean rates. What is more, setting t_{ratio} equal to 1.65 covers over the 94.80% of the runs (see Table 9). In spite of the range (1.65, 2.00] contains only about the 4.20%, which could be interpreted as outliers too, we finally adopt $t_{ratio} \leftarrow 2.00$, covering the 99.06% of runs. Such an election does not favor HVcMO20a

at all, quite the opposite. That fact supports the soundness of the later estimation of the times needed by HVcMO20a and by HillVallEA19a to execute the entire evolutionary optimization, not only particular iterations, as thus far. At that moment, the actual coefficients of the time functions of the algorithms are not relevant, but the t_{ratio}. Hence, for an input of n solutions they are assumed as:

$$t_{HillVallEA19}(n) \leftarrow n^2 \tag{20}$$

$$t_{HVcMO20a}(n) \leftarrow 2n^2 \tag{21}$$

Table 9. Number and percentage of runs covered by possible ratios of time, considering 5000 and 25,000 runs: 50 per each of 20 functions with populations of 5 sizes, at 5 levels of tolerance.

	Using a Tolerance of 1.0×10^{-5}		**Using All Levels of Tolerance**	
t_{ratio}	**Runs Covered**	**% Covered**	**Runs Covered**	**% Covered**
1.00	327	6.54	1576	6.30
1.40	3714	74.28	18,293	73.17
1.65	4745	94.90	23,714	94.86
1.75	4852	97.04	24,231	96.92
2.00	4953	99.06	24,765	99.06
2.50	4993	99.86	24,982	99.93
4.00	5000	100.00	25,000	100.00

6.3.3. Population Size and Execution Time

In Appendix B, Table A4 discloses the new analytical data about HVcMO20a and HillHallEA19, averaged over the same 50 runs per algorithm for each test problem, using tolerance 1.0×10^{-5}, whose results were studied in Sections 6.1 and 6.2. The first of those extra metrics is the final population size, which indicates the biggest population sampled, i.e., that of the last iteration. In addition, the total population size is the sum of the sizes (n) of all the sampled populations:

$$TotalPopSize \leftarrow \frac{\sum_{run=1}^{50} \sum_{i=1}^{\#Iterations_{run}} n_i}{50} \tag{22}$$

Lastly, adhering Equations (20) and (21), the overall execution time units are estimated as:

$$Time_{HillVallEA19} \leftarrow \frac{\sum_{run=1}^{50} \sum_{i=1}^{\#Iterations_{run}} {n_i}^2}{50} \tag{23}$$

$$Time_{HVcMO20a} \leftarrow \frac{\sum_{run=1}^{50} \sum_{i=1}^{\#Iterations_{run}} 2{n_i}^2}{50} \tag{24}$$

The inspection of these scores leads to an important discovery: HillVallEA19 is faster than HVcMO20a, in 9 of the 10 problems with search space volumes either small or large (marked in red, in Table 2), while HVcMO20a performs faster than HillVallEA19, in 9 out of 10 problems with medium volumes (indicated in blue). Such a volume seems to be key for the time demanded to approximate the problems. Figure 7 offers a graphical notion of this, considering average values.

This information is completed with the Wilcoxon test (see Table 10). The populations processed by HVcMO20a are significantly shorter than those handled by HillVallEA19. That confirms the first hypothesis in Section 4.2, that extra pre-niching search operators (like those of VMO) may reduce the total population in HillVallEA. A direct interpretation is that in HVcMO20a, the size of the sampled population increases less often than in HillVallEA19. The reduction of the number of solutions is more drastic with respect to the problems with a medium volume of the box-bounded search space. In consequence, HVcMO20a is estimated to run statistically faster than HillVallEA19, in that case.

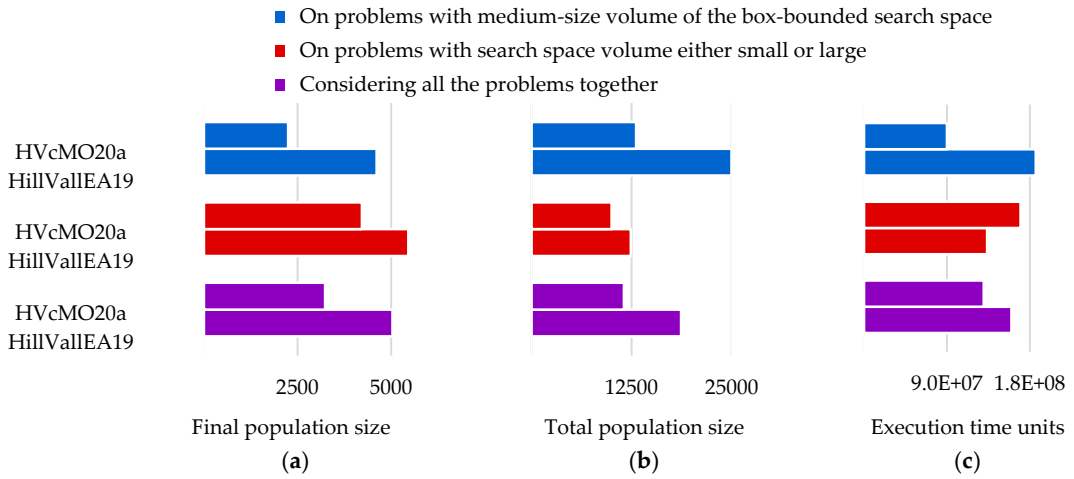

Figure 7. Mean results over the problems grouped by the size of the search space volume: (**a**) Final size of the population (actual); (**b**) Total size of the population (actual); (**c**) Execution time (estimate).

Table 10. Wilcoxon test as regards (final and total) population sizes and execution time. R− is the sum of ranks of comparisons where HVcMO20a outdoes HillVallEA19, and R+ is the opposite sum. If the W statistic is equal or less than the critical value, the null-hypothesis is rejected.

Volume of Search Space	Final Pop Size					Total Population Size					Execution Time Units				
	R+	R−	W	Critical Value	H_0	R+	R−	W	Critical Value	H_0	R+	R−	W	Critical Value	H_0
medium	0	55	0	8	Rejected	0	55	0	8	Rejected	1	54	1	8	Rejected
small + large	1	27	1	2	Rejected	3	42	3	6	Rejected	47	8	8	8	Rejected
all sizes	1	152	1	35	Rejected	3	187	3	46	Rejected	85	125	85	52	Accepted

Moreover, considering the validation suite entirely, HVcMO20a also executes more rapidly than HillVallEA19, not significantly though. That fact is influenced by the difference between populations, when it comes to problems with either small or large search space volume. The total number of solutions in HVcMO20a is significantly smaller than in HillVallEA19. In spite of that, such a reduction is not large enough to decrease properly the overall running time by HVcMO20a. Thus, HillVallEA19 is indeed estimated to execute significantly faster than HVcMO20a over that group of methods.

The second hypothesis stated in Section 4.2 is then partially verified as well, since the core part of it guesses that when the decrease of the total population size is sufficiently big, the overall running time of HillVallEA lessens. The remain of the hypothesis claims that the multimodal power of the algorithm keeps similar in those cases. As examined, that happens both over the whole test suite and over the problems having medium search space volumes only. With the comparison of HVcMO20a and HillVallEA19 in Section 6.2, it was already demonstrated that the performance of HillVallEA remains quite the same after incorporating the VMO search mechanisms, considering all the benchmark problems. It is proved below for those with medium-volume search space.

The fourth research objective of this paper is then accomplished, once HVcMO20a is a yet effective version of HillVallEA, also faster in several cases. The last goal of this investigation concerns the need for some criterion to select either HVcMO20a or HillVallEA19 to deal with new multimodal problems in future. If the box-bounded search space of the target problem has a medium-size volume, we recommend to use HVcMO20a; otherwise, HillVallEA19 is preferred. Our suggestions take into account the analyses on the execution time and about the main performance measures, which is completed (in Table 11) by verifying that in both situations, the algorithms are similarly effective.

Table 11. Scores averaged at all accuracy levels on problems grouped by the search space volume.

	Volumes of Medium Size		Volumes Small or Large	
Mean	**HillVallEA19**	**HVcMO20a**	**HillVallEA19**	**HVcMO20a**
PR	0.962	0.952	0.819	0.818
F1	0.979	0.973	0.887	0.887
dynF1	0.915	0.906	0.850	0.847

7. Conclusions and Future Work

The VMO-N framework, with its corresponding instances, and the NVMO algorithm, constitute the state-of-the-art regarding the multimodal optimization approaches for the VMO metaheuristic. The contrasts between them are examined in this study, and VMO-N is revised, turning into a very flexible scheme that can be vastly instantiated by incorporating any niching technique and further search strategies. Actually, its first competitive version is presented as HVcMO, specifically referred as HVcMO20a in the current setup. This newly launched algorithm outperforms not only the former instances of VMO-N, but also the NVMO method and several prominent multimodal optimizers, in some cases, in a statistically significant way.

At the same time, HVcMO20a is an extension of HillVallEA19, the ultimate version of the successful HillVallEA multimodal optimizer, whose main drawback concerns the use of very large populations. That limitation is tackled in this paper, since HVcMO20a reduces the number of solutions needed to approximate the studied benchmark problems (compared to HillVallEA19), which signifies a decrease of the overall execution time when the reduction of the population is sufficiently big. The experiments confirmed the mutual benefit of VMO and HillVallEA. The recent HVcMO borrows mainly the HVC niching method and the AMaLGaM-Univariate local optimizer from HillVallEA, resulting in a competitive algorithm for multimodal optimization. Additionally, the application of the search operators of VMO within HillVallEA makes it work significantly more express over certain problems. A practical advice derives for problems whose box-bounded search spaces are not larger than 10^{20}. If that volume is medium-size, i.e., in $(90,\ 10^3]$, it is recommended to employ HVcMO20a. Conversely, HillVallEA19 should be applied if the box-bounded search space is any small or large, i.e., if its volume is either in $(0, 90]$, or in $(10^3,\ 10^{20}]$.

This study provides multiple statistical evidences for the empirical comparison of the methods. Because such a practice is frequent in other areas of artificial intelligence but not in multimodal optimization, that is also an attempt to throw light on the utility of the non-parametric analysis for the research on this field. Besides, this investigation opens new research avenues for both HillVallEA and the VMO-N framework, as several interrogations remain unresolved. Beyond those of VMO, what advantage may other evolutionary search operators bring for HillVallEA? How beneficial is the late shrinkage of the population in HVcMO? When it comes to the local expansion within HVcMO, may other alternatives to decide the universe cause better effects? What is the impact of the frontier operator on the performance of such a new VMO-based multimodal optimizer? How can HVcMO and HillVallEA develop into more efficient and more effective algorithms? How well do they perform in real-world scenarios, and how well on problems with search spaces larger than those in this study? These questions determine some of the directions of forthcoming research.

Supplementary Materials: The following are available online at http://www.mdpi.com/2227-7390/8/5/665/s1, The source code of HVcMO20a that extends HillVallEA19 is available online [57], together with their 2000 output files (50 runs × 20 test problems × 2 algorithms), in the format requested in [28].

Author Contributions: Conceptualization, R.N. and C.H.K.; methodology, R.N. and C.H.K.; software, R.N.; validation, R.N. and C.H.K.; formal analysis, R.N.; investigation, R.N.; resources, C.H.K.; data curation, R.N.; writing—original draft preparation, R.N.; writing—review and editing, R.N. and C.H.K.; visualization, R.N.; supervision, C.H.K.; project administration, C.H.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Acknowledgments: This work was in part supported by the Japanese Government (MEXT).

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. PR, F1 and dynF1

Table A1. Mean values measured over 50 runs by ANBNWI-DE, HillVallEA19 and HVcMO20a for each test problem, assessed at all levels of accuracy.

Problem	ANBNWI-DE			HillVallEA19			HVcMO20a		
Id	PR	F1	dynF1	PR	F1	dynF1	PR	F1	dynF1
1	1.000	1.000	0.974	1.000	1.000	0.995	1.000	1.000	0.993
2	0.998	0.998	0.972	1.000	1.000	0.989	1.000	1.000	0.986
3	1.000	0.667	0.650	1.000	1.000	0.994	1.000	1.000	0.991
4	0.996	0.996	0.952	1.000	1.000	0.976	1.000	1.000	0.973
5	1.000	1.000	0.968	1.000	1.000	0.984	1.000	1.000	0.983
6	1.000	1.000	0.625	1.000	1.000	0.967	1.000	1.000	0.956
7	0.965	0.977	0.871	1.000	1.000	0.966	1.000	1.000	0.963
8	0.925	0.960	0.587	0.979	0.989	0.806	0.974	0.986	0.797
9	0.692	0.812	0.555	0.968	0.984	0.810	0.951	0.975	0.803
10	0.999	0.999	0.981	1.000	1.000	0.982	1.000	1.000	0.980
11	0.995	0.968	0.885	1.000	1.000	0.983	1.000	1.000	0.981
12	0.925	0.947	0.649	1.000	1.000	0.964	1.000	1.000	0.958
13	1.000	0.986	0.866	1.000	1.000	0.963	1.000	1.000	0.948
14	0.889	0.922	0.787	0.923	0.958	0.881	0.847	0.913	0.851
15	0.738	0.848	0.623	0.750	0.857	0.837	0.750	0.857	0.829
16	0.683	0.804	0.662	0.694	0.817	0.791	0.667	0.800	0.783
17	0.666	0.797	0.486	0.750	0.857	0.815	0.750	0.857	0.803
18	0.667	0.800	0.633	0.667	0.800	0.765	0.667	0.800	0.762
19	0.550	0.708	0.464	0.595	0.743	0.660	0.608	0.753	0.659
20	0.402	0.555	0.347	0.483	0.650	0.529	0.488	0.655	0.528

Appendix B. Ratio of Time and Population Size

Table A2. Individual values of ratio of means (RoM) and mean of ratios (RoM), for 50 runs and 250 runs of HVcMO20a and HillVallEA19 over each test problem with populations of different sizes.

	$n \leftarrow 2^6$		$n \leftarrow 2^7$		$n \leftarrow 2^8$		$n \leftarrow 2^9$		$n \leftarrow 2^{10}$		**All Pop Sizes**	
Problem	**RoM**	**MoR**	**RoM**	**MoR**	**RoM**	**MoR**	**RoM**	**MoR**	**RoM**	**MoR**	**RoM**	**MoR**
Id	**Considering 50 Runs per Function Using a Tolerance of 1.0×10^{-5}**											
1	1.343	1.354	1.346	1.351	1.330	1.330	1.342	1.342	1.321	1.321	1.325	1.340
2	1.250	1.281	1.363	1.369	1.301	1.304	1.334	1.342	1.319	1.319	1.321	1.323
3	1.305	1.327	1.375	1.382	1.324	1.324	1.354	1.354	1.316	1.316	1.324	1.341
4	1.166	1.189	1.287	1.293	1.282	1.283	1.340	1.340	1.311	1.311	1.314	1.283
5	1.196	1.221	1.309	1.315	1.294	1.296	1.350	1.350	1.313	1.313	1.318	1.299
6	1.362	1.382	1.440	1.451	1.459	1.460	1.373	1.373	1.312	1.312	1.335	1.396
7	1.311	1.330	1.333	1.344	1.339	1.340	1.332	1.332	1.308	1.309	1.315	1.331
8	1.480	1.525	1.427	1.442	1.488	1.489	1.430	1.431	1.372	1.372	1.393	1.452
9	1.616	1.635	1.454	1.460	1.417	1.419	1.356	1.357	1.296	1.300	1.325	1.434
10	1.093	1.108	1.211	1.215	1.260	1.260	1.334	1.334	1.319	1.319	1.314	1.247
11	1.124	1.141	1.233	1.238	1.276	1.278	1.355	1.356	1.326	1.326	1.322	1.268
12	1.406	1.431	1.451	1.471	1.404	1.406	1.380	1.380	1.315	1.315	1.339	1.401
13	1.265	1.304	1.390	1.397	1.358	1.359	1.397	1.397	1.341	1.341	1.353	1.360
14	1.296	1.313	1.389	1.398	1.438	1.444	1.486	1.487	1.380	1.381	1.403	1.405
15	1.372	1.443	1.536	1.562	1.465	1.475	1.468	1.469	1.386	1.391	1.412	1.468
16	1.266	1.316	1.256	1.288	1.341	1.350	1.418	1.420	1.386	1.386	1.378	1.352
17	1.265	1.309	1.306	1.338	1.439	1.455	1.443	1.446	1.388	1.389	1.396	1.387
18	1.049	1.126	1.020	1.068	1.083	1.098	1.200	1.208	1.233	1.236	1.185	1.147
19	1.094	1.141	1.073	1.114	1.131	1.147	1.201	1.211	1.237	1.240	1.194	1.171
20	1.010	1.112	0.939	0.991	1.000	1.036	1.097	1.123	1.182	1.186	1.088	1.090
Id	**Considering 250 Runs per Function: 50 Runs for Every Mesh Size Using the 5 Levels of Tolerance**											
1	1.370	1.387	1.347	1.352	1.360	1.360	1.343	1.343	1.333	1.334	1.336	1.355
2	1.318	1.342	1.351	1.355	1.339	1.342	1.332	1.338	1.341	1.341	1.339	1.344
3	1.371	1.394	1.336	1.343	1.362	1.363	1.331	1.332	1.342	1.342	1.341	1.355
4	1.236	1.254	1.287	1.291	1.329	1.331	1.328	1.328	1.337	1.337	1.333	1.308
5	1.202	1.229	1.311	1.316	1.331	1.335	1.337	1.337	1.328	1.328	1.329	1.309
6	1.394	1.422	1.438	1.448	1.470	1.472	1.364	1.364	1.335	1.336	1.350	1.408
7	1.358	1.378	1.344	1.352	1.344	1.345	1.310	1.312	1.324	1.326	1.323	1.343
8	1.480	1.527	1.451	1.465	1.492	1.494	1.414	1.415	1.385	1.385	1.400	1.457
9	1.609	1.626	1.450	1.457	1.432	1.434	1.342	1.343	1.317	1.318	1.338	1.436
10	1.142	1.156	1.200	1.204	1.287	1.290	1.316	1.317	1.331	1.331	1.321	1.259
11	1.137	1.158	1.211	1.217	1.309	1.311	1.337	1.338	1.344	1.344	1.334	1.273
12	1.408	1.437	1.445	1.464	1.446	1.449	1.360	1.360	1.343	1.343	1.356	1.411
13	1.270	1.308	1.368	1.374	1.392	1.394	1.373	1.375	1.355	1.356	1.360	1.361
14	1.269	1.288	1.389	1.399	1.474	1.479	1.464	1.465	1.395	1.397	1.411	1.405
15	1.340	1.392	1.498	1.523	1.495	1.503	1.453	1.454	1.407	1.409	1.424	1.456
16	1.254	1.301	1.249	1.274	1.355	1.367	1.399	1.401	1.388	1.389	1.376	1.346
17	1.263	1.308	1.274	1.301	1.425	1.443	1.414	1.418	1.402	1.402	1.396	1.374
18	0.956	1.047	1.029	1.098	1.103	1.120	1.198	1.206	1.232	1.235	1.182	1.141
19	1.022	1.051	1.011	1.049	1.134	1.150	1.185	1.195	1.248	1.250	1.187	1.139
20	1.002	1.100	0.945	0.993	1.023	1.064	1.079	1.104	1.187	1.192	1.089	1.091

Table A3. Number and percentage of runs covered by different ranges of time ratio.

	$n \leftarrow 2^6$	$n \leftarrow 2^7$	$n \leftarrow 2^8$	$n \leftarrow 2^9$	$n \leftarrow 2^{10}$	**All Sizes**	**Percentage Covered**		
$t_{ratio} \in$	**For Every Population Size: 50 Runs per Each of 20 Functions for Tolerance 1.0×10^{-5}**								
(0.00, 1.00]	161	89	53	21	3	327	6.54%		
(1.00, 1.20]	237	162	80	66	63	608	12.16%	}	27.52%
(1.20, 1.30]	150	183	243	55	137	768	15.36%		
(1.30, 1.40]	113	223	352	601	722	2011	40.22%	}	60.84%
(1.40, 1.65]	206	258	243	251	73	1031	20.62%		
(1.65, 1.75]	44	44	13	6	0	107	2.14%	}	4.16%
(1.75, 2.00]	57	29	13	0	2	101	2.02%		
(2.00, 2.50]	26	11	3	0	0	40	0.80%	}	0.94%
(2.50, 4.00]	6	1	0	0	0	7	0.14%		
#Runs	1000	1000	1000	1000	1000	5000			
$t_{ratio} \in$	**For Every Population Size: 50 per Each of 20 Functions Using the 5 Levels of Tolerance**								
(0.00, 1.00]	777	435	244	104	16	1576	6.30%		
(1.00, 1.20]	1185	882	386	354	273	3080	12.32%	}	26.43%
(1.20, 1.30]	686	942	710	591	598	3527	14.11%		
(1.30, 1.40]	631	1140	1958	2881	3500	10,110	40.44%	}	62.12%
(1.40, 1.65]	1035	1231	1512	1038	605	5421	21.68%		
(1.65, 1.75]	209	179	105	24	0	517	2.07%	}	4.21%
(1.75, 2.00]	313	140	68	7	6	534	2.14%		
(2.00, 2.50]	150	47	17	1	2	217	0.87%	}	0.94%
(2.50, 4.00]	14	4	0	0	0	18	0.07%		
#Runs	5000	5000	5000	5000	5000	25,000			

Table A4. Mean population size and running time over 50 runs with tolerance of 1.0×10^{-5}.

Problem	**Final Pop Size (Actual)**		**Total Pop Size (Actual)**		**Time Units (Estimated)**	
Id	**HVcMO20a**	**HillVallEA19**	**HVcMO20a**	**HillVallEA19**	**HVcMO20a**	**HillVallEA19**
1	65	64	137	128	18,186	8192
2	64	64	131	134	16,712	8602
3	64	64	128	128	16,384	8192
4	68	77	159	197	22,282	15,892
5	64	64	129	132	16,548	8438
6	302	461	1437	2198	777,748	883,671
7	1546	2929	7350	13,545	18,752,143	32,556,974
8	901	2191	7578	16,497	6,688,968	16,971,203
9	7946	19,988	85,183	165,129	700,629,722	1,529,669,468
10	68	70	187	223	25,887	15,892
11	252	358	868	1179	493,158	539,935
12	316	637	1230	2388	811,500	1,642,824
13	1341	1920	4429	6348	12,453,151	13,276,692
14	5714	9380	13,559	24,901	114,926,086	179,571,917
15	4096	8192	8893	17,299	45,454,131	90,031,473
16	6636	8192	13,509	16,721	128,647,938	90,269,860
17	4096	8192	8995	17,318	45,485,425	89,991,086
18	4096	4588	8504	9468	44,906,250	30,494,802
19	8028	8684	18,258	19,988	185,760,922	115,860,931
20	19,005	24,576	50,104	59,748	1,294,673,004	1,010,064,589

References

1. Reeves, C.R. Modern Heuristic Techniques. In *Modern heuristic search methods*; Rayward-Smith, V.J., Osman, I.H., Reeves, C.R., Smith, G.D., Eds.; John Wiley & Sons: New York, NY, USA, 1996; pp. 1–25.
2. Sörensen, K.; Sevaux, M.; Glover, F. A History of Metaheuristics. In *Handbook of Heuristics*; Martí, R., Pardalos, P.M., Resende, M.G., Eds.; Springer: Cham, Switzerland, 2018; pp. 1–18. [CrossRef]

3. Chica, M.; Barranquero, J.; Kajdanowicz, T.; Damas, S.; Cordón, Ó. Multimodal optimization: An effective framework for model calibration. *Inf. Sci.* **2017**, *375*, 79–97. [CrossRef]
4. Woo, D.-K.; Choi, J.-H.; Ali, M.; Jung, H.-K. A Novel Multimodal Optimization Algorithm Applied to Electromagnetic Optimization. *IEEE Trans. Magn.* **2011**, *47*, 1667–1673. [CrossRef]
5. Dilettoso, E.; Salerno, N. A Self-Adaptive Niching Genetic Algorithm for Multimodal Optimization of Electromagnetic Devices. *IEEE Trans. Magn.* **2006**, *42*, 1203–1206. [CrossRef]
6. Das, S.; Maity, S.; Qu, B.-Y.; Suganthan, P.N. Real-Parameter Evolutionary Multimodal Optimization-A Survey of the State-of-the-Art. *Swarm Evol. Comput.* **2011**, *2*, 71–88. [CrossRef]
7. Della Cioppa, A.; De Stefano, C.; Marcelli, A. Where Are the Niches? Dynamic Fitness Sharing. *IEEE Trans. Evol. Comput.* **2007**, *11*, 453–465. [CrossRef]
8. Kamyab, S.; Eftekhari, M. Using a Self-Adaptive Neighborhood Scheme with Crowding Replacement Memory in Genetic Algorithm for Multimodal Optimization. *Swarm Evol. Comput.* **2013**, *12*, 1–17. [CrossRef]
9. Sopov, E. Self-Configuring Ensemble of Multimodal Genetic Algorithms. In *Computational Intelligence. IJCCI 2015. Studies in Computational Intelligence, Vol 669*; Merelo, J.J., Rosa, A., Cadenas, J.M., Correia, A.D., Madani, K., Ruano, A., Filipe, J., Eds.; Springer: Cham, Switzerland, 2017; pp. 56–74. [CrossRef]
10. De Magalhães, C.S.; Almeida, D.M.; Barbosa, H.J.C.; Dardenne, L.E. A Dynamic Niching Genetic Algorithm Strategy for Docking Highly Flexible Ligands. *Inf. Sci.* **2014**, *289*, 206–224. [CrossRef]
11. Li, X. Niching Without Niching Parameters: Particle Swarm Optimization Using a Ring Topology. *IEEE Trans. Evol. Comput.* **2010**, *14*, 150–169. [CrossRef]
12. Nápoles, G.; Grau, I.; Bello, R.; Falcon, R.; Abraham, A. Self-Adaptive Differential Particle Swarm Using a Ring Topology for Multimodal Optimization. In Proceedings of the 13th International Conference on Intelligent Systems Design and Applications (ISDA'13), Bangi, Malaysia, 8–10 December 2013; pp. 35–40. [CrossRef]
13. Fieldsend, J.E. Running Up Those Hills: Multi-Modal Search with the Niching Migratory Multi-Swarm Optimiser. In Proceedings of the 2014 IEEE Congress on Evolutionary Computation (CEC'14), Beijing, China, 6–11 July 2014; pp. 2593–2600. [CrossRef]
14. Li, X. Developing Niching Algorithms in Particle Swarm Optimization. In *Handbook of Swarm Intelligence. Adaptation, Learning, and Optimization*; Panigrahi, B.K., Shi, Y., Lim, M.-H., Eds.; Springer: Berlin/Heidelberg, Germany, 2011; Volume 8, pp. 67–88. [CrossRef]
15. Qu, B.Y.; Suganthan, P.N. Novel Multimodal Problems and Differential Evolution with Ensemble of Restricted Tournament Selection. In Proceedings of the 2010 IEEE Congress on Evolutionary Computation (CEC'10), Barcelona, Spain, 18–23 July 2010; pp. 1–7. [CrossRef]
16. Epitropakis, M.G.; Plagianakos, V.P.; Vrahatis, M.N. Finding Multiple Global Optima Exploiting Differential Evolution's Niching Capability. In Proceedings of the 2011 IEEE Symposium on Differential Evolution (SDE'11), Paris, France, 11–15 April 2011; pp. 1–8. [CrossRef]
17. Thomsen, R. Multimodal Optimization Using Crowding-Based Differential Evolution. In Proceedings of the 2004 IEEE Congress on Evolutionary Computation (CEC'04), Portland, OR, USA, 19–23 June 2004; Volume 2, pp. 1382–1389. [CrossRef]
18. Epitropakis, M.G.; Li, X.; Burke, E.K. A Dynamic Archive Niching Differential Evolution Algorithm for Multimodal Optimization. In Proceedings of the 2013 IEEE Congress on Evolutionary Computation (CEC'13), Cancún, Mexico, 20–23 June 2013; pp. 79–86. [CrossRef]
19. Shir, O.M. Niching in Evolutionary Algorithms. In *Handbook of Natural Computing*; Rozenberg, G., Bäck, T., Kok, J.N., Eds.; Springer: Berlin/Heidelberg, Germany, 2012; pp. 1035–1069. [CrossRef]
20. Puris, A.; Bello, R.; Molina, D.; Herrera, F. Variable Mesh Optimization for Continuous Optimization Problems. *Soft Comput.* **2012**, *16*, 511–525. [CrossRef]
21. Navarro, R.; Falcon, R.; Bello, R.; Abraham, A. Niche-Clearing-Based Variable Mesh Optimization for Multimodal Problems. In Proceedings of the 2013 World Congress on Nature and Biologically Inspired Computing (NaBIC'13), Fargo, ND, USA, 12–14 August 2013; pp. 161–168. [CrossRef]
22. Navarro, R.; Murata, T.; Falcon, R.; Kim, C.H. A Generic Niching Framework for Variable Mesh Optimization. In Proceedings of the 2015 IEEE Congress on Evolutionary Computation (CEC'15), Sendai, Japan, 25–28 May 2015; pp. 1994–2001. [CrossRef]

23. Molina, D.; Puris, A.; Bello, R.; Herrera, F. Variable Mesh Optimization for the 2013 CEC Special Session Niching Methods for Multimodal Optimization. In Proceedings of the 2013 IEEE Congress on Evolutionary Computation (CEC'13), Cancún, Mexico, 20–23 June 2013; pp. 87–94. [CrossRef]
24. Maree, S.C.; Thierens, D.; Alderliesten, T.; Bosman, P.A.N. Real-Valued Evolutionary Multi-Modal Optimization Driven by Hill-Valley Clustering. In Proceedings of the 2018 Genetic and Evolutionary Computation Conference (GECCO'18), Kyoto, Japan, 15–19 July 2018; pp. 857–864. [CrossRef]
25. Bosman, P.A.N.; Grahl, J.; Thierens, D. Enhancing the Performance of Maximum–Likelihood Gaussian EDAs Using Anticipated Mean Shift. In *Parallel Problem Solving from Nature—PPSN X. PPSN 2008*; Lecture Notes in Computer Science; Rudolph, G., Jansen, T., Beume, N., Lucas, S., Poloni, C., Eds.; Springer: Berlin/Heidelberg, Germany, 2008; Volume 5199, pp. 133–143. [CrossRef]
26. Bosman, P.A.N.; Grahl, J.; Thierens, D. Benchmarking Parameter-Free AMaLGaM on Functions With and Without Noise. *Evol. Comput.* **2013**, *21*, 445–469. [CrossRef] [PubMed]
27. Maree, S.C.; Alderliesten, T.; Bosman, P.A.N. Benchmarking HillVallEA for the GECCO 2019 Competition on Multimodal Optimization. Available online: https://arxiv.org/abs/1907.10988v1 (accessed on 25 October 2019).
28. Competition on Niching Methods for Multimodal Optimization. Available online: http://www.epitropakis.co.uk/gecco2019/ (accessed on 23 October 2019).
29. Kanemitsu, H.; Imai, H.; Miyaskoshi, M. Definitions and Properties of (Local) Minima and Multimodal Functions using Level Set for Continuous Optimization Problems. In Proceedings of the 2013 International Symposium on Nonlinear Theory and its Applications (NOLTA2013), Santa Fe, NM, USA, 8–11 September 2013; pp. 94–97. [CrossRef]
30. Zhai, Z.; Li, X. A Dynamic Archive Based Niching Particle Swarm Optimizer Using a Small Population Size. In Proceedings of the This paper appeared at the Thirty-Fourth Australasian Computer Science Conference (ACSC2011), Perth, Australia, 17 January 2011; Reynolds, M., Ed.; Conferences in Research and Practice in Information Technology (CRPIT), Australian, Computer Society, Inc.: Perth, Australia, 2011; Volume 113.
31. Kronfeld, M.; Zell, A. Towards Scalability in Niching Methods. In Proceedings of the 2010 IEEE Congress on Evolutionary Computation (CEC'10), Barcelona, Spain, 18–23 July 2010; pp. 1–8. [CrossRef]
32. Puris, A.; Bello, R.; Molina, D.; Herrera, F. Optimising Real Parameters Using the Information of a Mesh of Solutions: VMO Algorithm. In Proceedings of the 2012 IEEE Congress on Evolutionary Computation (CEC'12), Brisbane, QLD, Australia, 10–15 June 2012; pp. 1–7. [CrossRef]
33. Pétrowski, A. Clearing Procedure as a Niching Method for Genetic Algorithms. In Proceedings of the IEEE Conference on Evolutionary Computation, Nagoya, Japan, 20–22 May 1996; pp. 798–803. [CrossRef]
34. Sareni, B.; Krähenbühl, L. Fitness Sharing and Niching Methods Revisited. *IEEE Trans. Evol. Comput.* **1998**, *2*, 97–106. [CrossRef]
35. Li, J.P.; Balazs, M.E.; Parks, G.T.; Clarkson, P.J. A Species Conserving Genetic Algorithm for Multimodal Function Optimization. *Evol. Comput.* **2002**, *10*, 207–234. [CrossRef] [PubMed]
36. Gan, J.; Warwick, K. Dynamic Niche Clustering: A Fuzzy Variable Radius Niching Technique for Multimodal Optimisation in GAs. In Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No.01TH8546), Seoul, Korea, 27–30 May 2001; Volume 1, pp. 215–222. [CrossRef]
37. Brown, M.S. *A Species-Conserving Genetic Algorithm for Multimodal Optimization*; Nova Southeastern University: Fort Lauderdale, FL, USA, 2010.
38. Iwase, T.; Takano, R.; Uwano, F.; Sato, H.; Takadama, K. The Bat Algorithm with Dynamic Niche Radius for Multimodal Optimization. In Proceedings of the 2019 3rd International Conference on Intelligent Systems, Metaheuristics & Swarm Intelligence, Malé, Maldives, 24 March 2019; pp. 8–13. [CrossRef]
39. Solis, F.J.; Wets, R.J.B. Minimization by Random Search Techniques. *Math. Oper. Res.* **1981**, *6*, 19–30. [CrossRef]
40. Li, X.; Engelbrecht, A.; Epitropakis, M.G. Benchmark Functions for CEC'2013 Special Session and Competition on Niching Methods for Multimodal Function Optimization. Technical Report. Evolutionary Computation and Machine Learning Group, RMIT University: Australia, 2013. Available online: https://titan.csit.rmit.edu.au/~e46507/cec13-niching/competition/cec2013-niching-benchmark-tech-report.pdf (accessed on 23 October 2019).
41. Qu, B.Y.; Liang, J.J.; Suganthan, P.N. Niching Particle Swarm Optimization with Local Search for Multi-Modal Optimization. *Inf. Sci.* **2012**, *197*, 131–143. [CrossRef]

42. Della Cioppa, A.; Marcelli, A.; Napoli, P. Speciation in Evolutionary Algorithms: Adaptive Species Discovery. In Proceedings of the 2011 Genetic and Evolutionary Computation Conference (GECCO'11), Dublin, Ireland, 12 July 2011; pp. 1053–1060. [CrossRef]
43. Ursem, R.K. Multinational Evolutionary Algorithms. In Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406), Washington, DC, USA, 6–9 July 1999; Volume 3, pp. 1633–1640. [CrossRef]
44. *Towards a New Evolutionary Computation. Studies in Fuzziness and Soft Computing*; Lozano, J.A.; Larrañaga, P.; Inza, I.; Bengoetxea, E. (Eds.) Springer: Berlin/Heidelberg, Germany, 2006; Volume 192. [CrossRef]
45. Dong, W.; Yao, X. NichingEDA: Utilizing the Diversity inside a Population of EDAs for Continuous Optimization. In Proceedings of the 2008 IEEE Congress on Evolutionary Computation (CEC'08), Hong Kong, China, 1–6 June 2008; pp. 1260–1267. [CrossRef]
46. Chen, B.; Hu, J. An Adaptive Niching EDA Based on Clustering Analysis. In Proceedings of the 2010 IEEE Congress on Evolutionary Computation (CEC'10), Barcelona, Spain, 18–23 July 2010; pp. 1–7. [CrossRef]
47. Yang, P.; Tang, K.; Lu, X. Improving Estimation of Distribution Algorithm on Multimodal Problems by Detecting Promising Areas. *IEEE Trans. Cybern.* **2015**, *45*, 1438–1449. [CrossRef] [PubMed]
48. HillVallEA. Available online: https://github.com/scmaree/HillVallEA (accessed on 28 October 2019).
49. Rodrigues, S.; Bauer, P.; Bosman, P.A.N. A Novel Population-Based Multi-Objective CMA-ES and the Impact of Different Constraint Handling Techniques. In Proceedings of the 2014 Genetic and Evolutionary Computation Conference (GECCO'14), Vancouver, BC, Canada, 12–16 July 2014; pp. 991–998. [CrossRef]
50. Wolpert, D.H.; Macready, W.G. No Free Lunch Theorems for Optimization. *IEEE Trans. Evol. Comput.* **1997**, *1*, 67–82. [CrossRef]
51. Navarro, R. *Optimización Basada En Mallas Variables Con Operador de Fronteras Basado En Búsqueda Genética (Variable Mesh Optimization with Frontiers Operator Based on Genetic Search)*; Universidad de Holguín: Piedra Blanca, Holguín, Cuba, 2012; Available online: https://repositorio.uho.edu.cu/jspui/handle/uho/444 (accessed on 10 March 2020).
52. Preuss, M. Improved Topological Niching for Real-Valued Global Optimization. In *Applications of Evolutionary Computation. EvoApplications 2012. Lecture Notes in Computer Science*; Springer: Berlin/Heidelberg, Germany, 2012; Volume 7248. [CrossRef]
53. Ahrari, A.; Deb, K.; Preuss, M. Multimodal Optimization by Covariance Matrix Self-Adaptation Evolution Strategy with Repelling Subpopulations. *Evol. Comput.* **2017**, *25*, 439–471. [CrossRef] [PubMed]
54. CEC2013. Available online: https://github.com/mikeagn/CEC2013 (accessed on 15 January 2020).
55. Demšar, J. Statistical Comparisons of Classifiers over Multiple Data Sets. *J. Mach. Learn. Res.* **2006**, *7*, 1–30.
56. Derrac, J.; García, S.; Molina, D.; Herrera, F. A Practical Tutorial on the Use of Nonparametric Statistical Tests as a Methodology for Comparing Evolutionary and Swarm Intelligence Algorithms. *Swarm Evol. Comput.* **2011**, *1*, 3–18. [CrossRef]
57. HVcMO. Available online: https://github.com/ricardonrcu/HVcMO (accessed on 24 March 2020).