Multimodal Optimization: Formulation, Heuristics, and a Decade of Advances

Mike Preuss, Michael Epitropakis, Xiaodong Li, and Jonathan Fieldsend

Abstract Multimodal optimization is a relatively young term for the aim of finding several solutions of a complex objective function simultaneously. This has been attempted under the denomination 'niching' since the 1970s, transferring ideas from biological evolution in a very loose fashion. In this chapter we more formally define it, and then highlight its most important perspectives: how do we measure what is good? On what problems do we measure it? Which type of algorithms may be effectively employed for multimodal optimization? How do they relate to each other? Competitions at two major evolutionary computation conferences have driven algorithm development in recent years. We therefore report, in a concise fashion, what we have learned from competition results and give an outlook on interesting future developments.

1 Introduction

What is the essence of *multimodal optimization* (MMO) when lots of mathematical optimization methods have been used to deal with multimodal problems since the 1970s and even before? Experience shows that real-world problems are very often too complex to be treated successfully with classical optimization methods. This is certainly a biased view, as simple problems are already solvable by means of standard optimization methods, and only the 'leftovers' are difficult enough to require some reasoning on how to treat them. These non-trivial real-world problems are almost always multimodal, either by nature of the underlying function or enforced by the islands of feasibility that are induced by constraints, or both. Although it is definitively as interesting as it is important to also treat constrained multimodal optimization, we abstain from doing so as this area is not that well explored yet and thus much more speculative.

Here, we provide an introduction and overview of the state-of-the-art of (unconstrained) multimodal optimization, primarily, but not exclusively, from an evolutionary computation (EC) perspective. Interestingly, many of the ideas and techniques that emerged for multimodal optimization in the EC field in recent years have predecessors in global optimization. Some of these have been forgotten for some decades and only recently been brought to light again (see e.g. [1, 2]). We may therefore speak of *convergent evolution* in this case, as obviously the need to overcome a specific set of difficulties induced similar ideas in several research areas.

Let us first attempt to non-formally approach the meaning of the term ‘multimodal optimization’. In our view, it is an umbrella term that contains several types of applications of specialized optimization methods to multimodal functions, but always with the focus on eventually delivering multiple solutions. As previously stated, very many application problems are multimodal in nature. However, they are often treated by simpler methods of convex optimization [3] that make strong assumptions on the geometrical shape of the function, or by methods from global optimization [2, 4] that are designed to find *one* global optimum of multimodal optimization problems. Why shall we be interested in determining several optima, and possibly some that are not even globally optimal? Li, *et al.* [5] discussed this issue nearly 20 years ago and gave two main reasons:

- to increase the chance of actually locating the global optimum, and
- to provide insights into the problem and suggest alternative innovative solutions.

One may argue that the latter argument is actually two arguments, as understanding optimization problems in order to better adapt algorithms to them, is of value even if no alternative solutions are desired. The other half of the argument reflects practitioners’ difficulties to accurately model a problem. The provided global optima may actually be infeasible (see [6] for an example), or cannot be realized exactly (i.e. when only a discrete set of shapes/parts/sizes is available, or machining is limited to a particular fidelity). The model itself may be in error, globally, or locally, meaning a range of disparate (but predicted high quality) solutions is of considerable value to mitigate the risk when the realizing the design(s) in practice. Besides, we may also encounter the situation that the decision-maker rejects specific solutions for reasons that have never been seen as part of the model, either because they have not been expected to be relevant, or because they are very hard to be expressed mathematically and are rather a matter of taste (so-called soft constraints). An example that is somewhere between both cases is the sound of a car combustion engine that is optimized for fuel efficiency but should still sound somewhat typical (not too different from the expectation of the customer).

Once we accept that searching for multiple optima at once is sometimes required, the next question is then how to define the properties of the *particular* optima we are interested in discovering. This becomes increasingly important if there are a large number of potential optima in an optimization problem. Even if we insist on returning global optima in our solution set, we still have at least three choices [1]:

1. find all ‘best’ solutions, that is all points in the search space that are globally optimal, or
2. obtain all optima, global or not, possibly better than a specified threshold, or
3. find at least one globally optimal solution.

Taking into account that for most practical black-box problems, we do not know the value of the global optimum and the problem to decide if a point is a global optimum is thus as hard as solving the optimization problem itself, we may further relax the requirements and just ask for some ‘good’ optima. However, this does not help much, because there is no clear criterion on which ones to look for if there are many, how to balance diversity and quality. The only thing that is clear is that the sought optima shall be distant enough from each other to actually make a difference in their practical implementation. This requires what has been very early termed as ‘diversity maintenance’, albeit also a fuzzy term.

We have come a long way already with niching and closely related similar approaches from global optimization, but the reshaped and unified field of *multimodal optimization* is still forming. We aim to consolidate it here and also want to point to interesting developments as well as open problems.

However, this work is not a survey. For that, you may consider consulting one of [7, 1, 8]. Here, we want to point to the important lines of scientific development, not to every single attempt, and we largely do so from the perspective of the multimodal optimization competitions of the last years.

The chapter now continues with a definition of the optimization task before we turn to the available performance measures in Sec. 3. Section 4 then deals with test problems and generators, and Sec. 5 provides an overview over common algorithmic approaches. We report on the outcome of recent years competitions in Sec. 6, and then conclude the chapter with an outlook to possible future research directions.

2 Definitions

We now outline the multimodal optimization task more formally, starting by disambiguating it from the standard optimisation task.

2.1 The general optimisation problem

The general optimization task is to find the 'best-performing' design, $\mathbf{x}^\star$, from some search space (or *domain*) $\mathcal{X}$, $\mathbf{x}^\star \in \mathcal{X}$, given some cost function, f. Without loss of generality, maximization problems can be converted to minimization variants though the simple multiplication of -1. $\mathcal{X}$ may be in a continuous space, a discrete space, a permutation space, mixed integer space, etc. For the use of our illustrations here, $\mathcal{X}$ is a boxed-constrained continuous space, i.e. $\mathcal{X} = [\boldsymbol{\ell}, \boldsymbol{u}] \subset \mathbb{R}^n$. $n \in \mathbb{N}$ is the fixed (in this case) number of decision variables. The vectors $\boldsymbol{\ell} = (\ell_1, \ldots, \ell_n)^\top$ and $\boldsymbol{u} = (u_1, \ldots, u_n)^\top$ are called the lower and upper bounds of $\mathcal{X}$, respectively.

There may be other constraints for the problem, which may be expressed as equality or inequality functions of $\mathbf{x}$, $g_i(\mathbf{x})$, however in this chapter we shall focus exclusively on continuous box-constrained problems — which reflects the properties of the test problems in the competition results we discuss in later sections.

2.2 The multimodal optimization problem

The combination of $\mathcal{X}$, f and $\mathbf{g}$ together are sufficient to describe the general optimization task, however in the *multimodal* optimization task we are also concerned with a *neighborhood-function*, which is used to characterize the local search landscape features.

For optimization problems in the continuous space, a common neighborhood-function is $N(\boldsymbol{y}) = \{\boldsymbol{x} \in \mathcal{X} \mid d(\boldsymbol{x}, \boldsymbol{y}) \leq \epsilon\}$ which describes a ball-shaped neighborhood of a point $\boldsymbol{y} \in \mathcal{X}$. Using neighborhood-function N, and an $\epsilon > 0$, we can say $\boldsymbol{x}'$ is a local minimum at radius ϵ if $\nexists \boldsymbol{x} \in N(\boldsymbol{x}') : f(\boldsymbol{x}) \leq f(\boldsymbol{x}')$. on the definition provided in in [9, p. 6], the multimodal optimization task may be formulated explicitly as:

Definition 1 (Multimodal minimization problem). Let there be ν local minima $\mathbf{x}_1^*, \dots, \mathbf{x}_\nu^*$ of f in $\mathcal{X}$. If the ordering of these optima is $f(\mathbf{x}_1^*) \le \dots \le f(\mathbf{x}_l^*) < h \le \dots \le f(\mathbf{x}_\nu^*)$, a multimodal minimization problem is given as the task to approximate the set $\bigcup_{i=1}^{l} \{\mathbf{x}_i^*\}$.

The variable h in this definition is simply a threshold to potentially exclude some of the lower quality optima. Conversely, $h = \infty$ indicates we are interested in discovering *all* local optima of a problem. If $h = f(\mathbf{x}_1^*)$, we are only interested in approximating (all) the global optima.

Let P be the obtained approximation set. Additional constraints may be applied to this set, to obtain more specific problem definitions. For example, the cardinality of P could be restricted by requiring $|P| \le k$. If $k = 1$, we have the conventional global optimization problem, where typically only one solution is sought. Another issue relates to the diversity requirements, which could be formulated by demanding $\forall \boldsymbol{x}, \boldsymbol{y} \in P, \boldsymbol{x} \ne \boldsymbol{y} : d(\boldsymbol{x}, \boldsymbol{y}) > \kappa$, i. e., the distance between any two solutions may not be smaller than some threshold κ. Figure 1 illustrates the desired set for various h, ϵ and κ on a multi-modal test problem from [10]. Alternatively, a more sophisticated diversity measure on the set P could be calculated, which may, for instance, lead to a multi-objective formulation of the problem [11].

Given the above, one can see there are a range of different ways the multi-modal optimization task make be defined, which unsurprisingly means there are a number of different (aligned) ways that the degree of *success* in solving such a task may be measured. We now outline some of these performance measures.

3 Performance Measures

Measuring performance in multimodal optimization has additional challenges beyond those exhibited by standard global optimization, as the definition of goodness is far more complex. The first striking difference is that as in multi-objective optimization, measurements are always computed from sets, not from single solutions [12, 11]. Furthermore, it is necessary to know where the sought optima are, or at least how good they are. The three principal dimensions of measuring performance in multimodal optimization are: quality, diversity, and speed.

Depending on the application context, any of these may be more important than the others. Generally, this means that we cannot expect to find one measure that captures all our criteria, the measuring process is ambiguous by design and we have to make decisions regarding what we consider as the most important features of a measure, and indeed their relative importance. Leaving the temporal aspect (when are solutions found) aside for the moment, and starting from all search points (solutions) an algorithm has seen doing one run, this leads to a multi-step process:

subset selection	remove all solutions that do not meet the minimum performance threshold
attribution	decide which optima each solution belongs to
assignment	provide performance values for each of the single known optima/basins and their associated solutions
aggregation	compute an overall performance value on the basis of the single values

Thus, apart from the detection of multiple local or global optima, also an explicit identification or subset selection of solutions that are needed for measuring within the possibly large set of explored solutions is necessary (see Fig. 2). Note that for global optimization, this is usually implicitly done by just selecting the best evaluated solution.

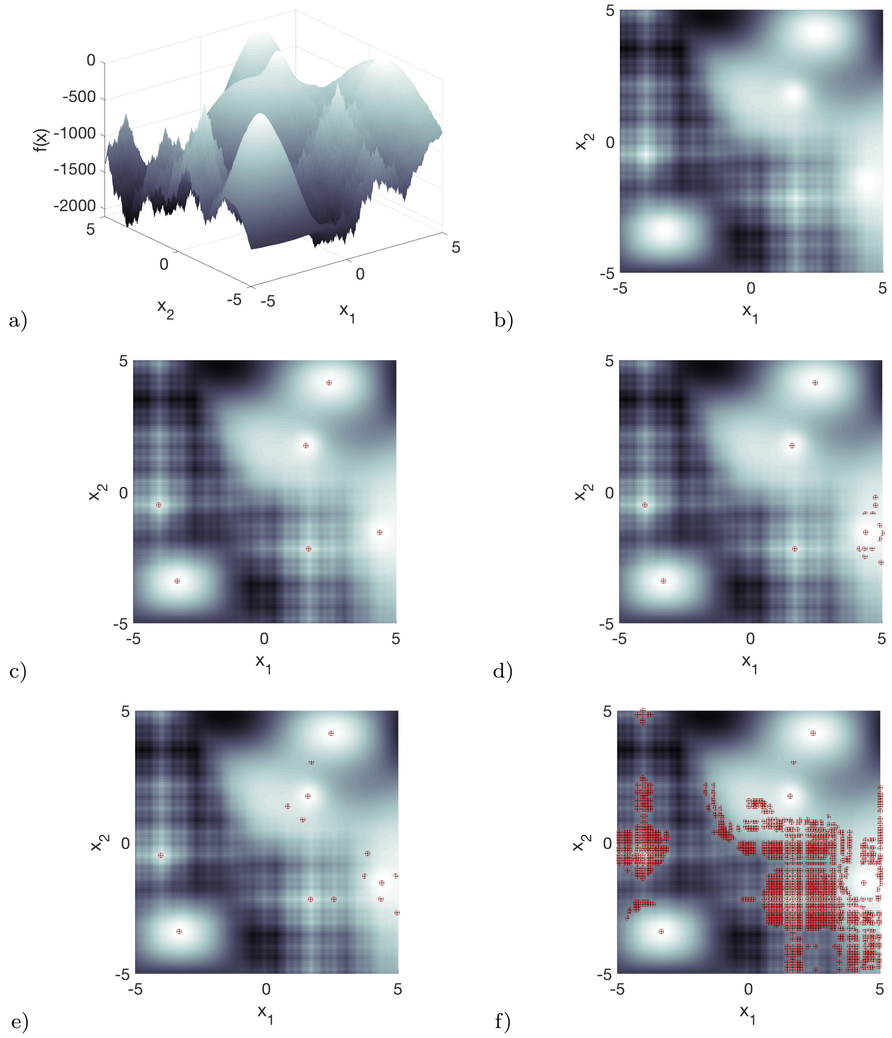

Fig. 1 Illustration of different targets under the multimodal optimization task. Composite Function 1 used from [10] with $n = 2$. Panels (a) and (b) illustrate the problem in three dimensions and the plane. Panel (c) has the desired solutions marked for $h = 0$, $\epsilon = 0.02$ and $\kappa = 0$ (the global optima), with $\kappa \gtrapprox 9.56$ a single global optima is satisfactory. Panel (d) has the desired solutions for $h = -200$, $\epsilon = 0.02$ and $\kappa = 0.2$ (the global optima and some high quality local optima). Panel (e) has the desired solutions for $h = -500$, $\epsilon = 0.02$ and $\kappa = 0.5$ (the global optima and some high quality local optima which are distant from them). Panel (f) has the desired solutions for $h = -1000$, $\epsilon = 0.02$ and $\kappa = 0.1$ (all global optima, and many local optima of moderate performance and better).

Ideally, the optimization algorithms will already perform this task. However, not all algorithms provide this explicit identification. Sometimes, it is easy to add, e. g., when the algorithm relies on several convergent local searches. Then, we can simply collect the best solutions of the local searches, and remove the duplicates. For some population-based approaches, however, the whole

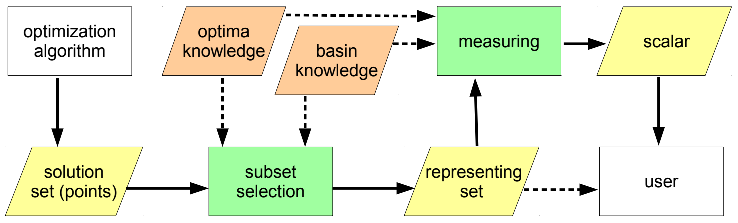

Fig. 2 How to evaluate performance of a multimodal optimization algorithm, general view (from [11]).

history of evaluations may have to be searched for suitable solutions. This might require quadratic-time algorithms for a reasonable identification, which can get computationally expensive. Also, depending on the concrete measures, it may be necessary to use knowledge on the location of optima and/or basins for the selection process. However, without subset selection, multimodal optimization algorithms are not very useful in practice as they emit huge solution sets the decision maker then has to choose from.

In the following, we list some possible measures and discuss briefly their advantages and disadvantages, mostly based on [11]. The first group of indicators is purely statistical and does not need any prior knowledge of a problem. It corresponds to the diversity goal only. In order to measure diversity of a solution set, one may use the sum of (all) distances, or the sum of nearest neighbor distances. The latter may be better at avoiding clustering of solutions. Solow-Polasky diversity is a biologically motivated diversity measure that has some advantages, but is computationally expensive and thus suitable for rather smaller solution sets. Additionally, it uses a critical parameter, which is not trivial to specify. For more details on these measures please see [9].

Given the prior knowledge of the optima, i.e., the global optima are known in advance, we can measure the peak ratio (PR) by checking for each desired optimum if there is a point in the solution set that approximates it closely enough. This performance measure also requires a parameter in order to define how 'close' it is from an known optimum. By measuring the peak distance, that is the average distance from each optimum to the nearest point in the solution set, we can avoid that parameter. This measure is related to the inverted generational distance well known in multi-objective optimization. We can generalize it even further and define an averaged Hausdorff distance, which is especially suitable for comparisons of different solution set sizes as it penalizes unnecessary points. Whereas these three measures only account for the search space distances to the sought optima, we can also use the objective space distance, that is the average error per peak, in the peak inaccuracy (originally designed as peak accuracy) measure. Similarly, with knowledge of the basins of attraction, we can also define basin ratio and basin inaccuracy. However, as basin information is much harder to obtain than optima information, these measures are hardly ever used. This group of measures corresponds to the quality goal.

We have seen that there are many different ways to measure performance of a multimodal optimization algorithm and there is no optimal choice. For that reason, the competitions on niching methods for multimodal optimization held at different evolutionary computation conferences in the last years use a combination of three performance criteria:

Peak Ratio (PR): This measure is kept for comparability to earlier results and is used as described in [10]. It measures the average percentage of the identified known global optima by an algorithm over multiple execution runs. In detail, given the fixed number of maximum function evaluations (MaxFEs) and the desired level of accuracy (ϵ), it can be calculated according to the following equation:

$$PR = \frac{\sum_{i=1}^{NR} GO_{\alpha}^{i}}{GO * NR}, \tag{1}$$

where GO_{α}^{i} denotes the number of global optima found at the end of the i-th execution run by algorithm α, GO the number of known global optima, and NR the number of runs. Note that we do not do subset selection for this measure, the whole archive of sampled solutions in one run can be submitted and is searched for points that are near to the known (global) optima. This is also the recall (sensitivity) value for the other two criteria.

Static F_1: Here we are not only interested in the ability of the algorithm to find all sought optima, but also to do proper subset selection and deliver only the important portion of the solution set. In information retrieval, this is called precision (the fraction of useful information), and its product with the recall (the fraction of the total amount of relevant information that was actually retrieved) results in the F_1 measure.

More specifically, given an algorithm α, and a problem instance, let GO be the number of all global optima for that problem. Let $S\alpha$ be the set of potential solutions that have been obtained by algorithm α and let GO_{S_α} be the number of distinct global optima that exist in $S\alpha$. The precision metric denotes the fraction of retrieved data that are useful, where under the MMO formulation this can be translated as the fraction of the number of identified global optima by algorithm α over the number of solutions identified by α, i.e., $precision = GO_{S_\alpha}/|S_\alpha|$, where $|\cdot|$ denotes the cardinality of a set. Similarly, recall denotes the fraction of the total amount of relevant data that were actually retrieved, which in our context means the fraction of the number of global optima an algorithm α identifies over the number of existing global optima for a given problem $recall = GO_{S_\alpha}/GO$. It can be clearly observed that the recall measure essentially defines the peak ratio (PR) of an algorithm α. The static F_1 measure can be calculated according to the following equation:

$$F_1 = 2 \times \frac{precision \times recall}{precision + recall} = \frac{2 \times GO_{S_\alpha}}{GO + |S_\alpha|} \tag{2}$$

Just as the simple PR measure, the F_1 value is bounded by 1. It resembles the harmonic mean of precision and recall and takes its best value if all sought optima are found, and only these. Both measures correspond to the goals quality and diversity, but not to speed.

Dynamic F_1: The efficiency of an algorithm in terms of speed (how fast it identifies the desired global optima) can not be captured by the above mentioned metrics. Therefore, we introduce a dynamic F_1 measure in order to separate slower from faster algorithms. This can be achieved if the point in time (evaluation number under a fixed budget) is known when each solution that ends up in the final solution set $|S_\alpha|$ has been obtained. We simply compute the static F_1 value for each point in time when the solution set changes, and then integrate over this value (over the number of evaluations allowed in one run). Thus, the dynamic F_1 is the area under the curve of the static version F_1 over time, and is also bounded by 1.

It is computed as follows: let the maximum budget of function evaluations of a problem instance be $MaxFEs$. Let GO_{S_α} be the solution set identified by algorithm α, ordered by the number of function evaluations f_i when solution i was added to the set, where $i \in [1, |GO_{S_\alpha}|]$. Then we can denote the subset of GO_{S_α} from solution i up to, including, solution j, where $i < j$ and $i, j \in [1, |GO_{S_\alpha}|]$, as $GO_{S_\alpha}(i, j)$. The dynamic F_1 measure ($dynF_1$) can be defined according to the following equation:

$$dynF_1 = \sum_{i=2}^{|GO_{S_\alpha}|} \left(\frac{f_i - f_{i-1}}{MaxFEs_p} F_1(GO_{S_\alpha}(1, i-1)) \right) + \frac{MaxFEs_p - f_{|GO_{S_\alpha}|}}{MaxFEs_p} F_1(GO_{S_\alpha}) \quad (3)$$

Intuitively the first term of the equation calculates the area under the curve of the static F_1 metric for a given solution set, i.e., $GO_{S_\alpha}(1, i-1)$, while the second term performs the last step of the integral to reach the maximum available budget. Larger values of the metric indicate faster conversion rates to the desired global optima set.

Please note that all these three measures are based on the peak ratio (PR) and therefore need information on the sought optima in order to be computed. Whereas diversity based measures discussed above do not need that, if used alone, but they can only provide a very limited view onto the multimodal optimization performance. Thus, it is much harder to reasonably measure performance for real-world test problem when optima information is not known and we very much depend on benchmarks in order to develop better algorithms.

4 Benchmark Suites and Problem Generators

The capability of a niching method in locating multiple optimal (or near optimal) solutions within a single optimization run can be assessed by using a series of test functions constructed explicitly with multimodality in mind, i.e., the fitness landscape as computed by the objective function needs to be multimodal. In such a scenario, a search algorithm equipped with a niching method is challenged to find as many as possible optimal (or near optimal) solutions in an optimization run. This is clearly more demanding than just finding a single global optimal solution, which is typical for a standard meta-heuristic algorithm. The earliest work on designing benchmark multimodal test functions was done by Deb in his 1989 master thesis [13]. Deb's test suite includes several simple 1 and 2-dimensional test functions, with multiple peaks of varying heights and distances between them. Subsequent to that, a more challenging test function with millions of local optima and 32 global optima was proposed by Goldberg *et al.* [14].

Test function generators have also been developed to construct multimodal test functions with more versatile and sophisticated sets of properties, also with the goal of being able to benchmark on a larger set of instances and not single problems. Gallagher and Yuan proposed such a generator based on the superposition of Gaussians [15]. Rönkkönen *et al.* [16] developed a more general multimodal test function generator using tunable function families, such as the *cosine* and *quadratic* function families. These test functions have more controllable numbers of global and local optima, scalable to higher dimensions. The functions can be rotated to a random angle, and each dimension of a function can be stretched independently using Bezier curves. Similarly, Singh and Deb proposed to use the *hump* function family to generate a series of multimodal test functions with different levels of complexity, e.g., several peaks can be generated at random locations with different shapes and sizes [17].

More complex multimodal fitness landscapes (i.e., with multiple global optima) can be constructed by superposition of several basic functions each with a single global optimum, as proposed by Qu and Suganthan [18]. Several such composition functions were included in the technical report "Benchmark Functions for CEC2013 Special Session and Competition on Niching Methods for Multimodal Function Optimization" [10]. Twenty multimodal test functions ranging from simple, separable, symmetric, and non-scalable to more challenging, non-separable, non-symmetric, and scalable were included in this benchmark suite, which has been used in the GECCO/CEC niching competition series running since 2013. Summarized results and rankings on many state-of-the-art niching methods are made publicly available (see details in a subsequent section on niching competition results). It has been widely adopted as the standard benchmark in evaluating niching methods in recent literature. The niching competition series aims to provide a common platform to facilitate meaningful evaluation and fair comparisons of different niching methods across a range of problem characteristics and difficulty levels.

Wessing [9] suggested the highly versatile polynomial based *Multiple Peaks Model 2* (MPM2) generator (in more detail described in [19]) that is based on former works by Preuss and Lasarczyk [20]. The most recent attempt in improving benchmark functions was carried out by Ahrari and Deb [21], where a general procedure for generating scalable multimodal test functions was proposed. This new procedure constructs a composite function through an application of two levels of basic functions, with the first level made up of two basic functions g_I and g_{II}, whose outputs in turn are fed into another basic function g_{III} in the 2nd level. Desirable properties such as ill-conditioning and variable interaction can be designed through manipulation of the adopted basic functions within the general framework. The composite functions generated are scalable to dimensions, with controllable number of global optima, variable shapes and sizes of basins. In addition, constrained multimodal test functions can be also constructed through introducing constraints in the basic function g_I. Note that the first attempt in designing constrained multimodal test functions was made by Deb and Saha in [22].

To evaluate niching methods in dealing with uncertainty, multimodal test functions with consideration of robustness were also developed by Alyahya, *et al.* [23]. Some of these robustness test functions were built upon the widely used standard multimodal benchmark, i.e., CEC'2013 niching competition technical report [10].

5 Popular algorithmic approaches and history of the field

Studies on niching methods can be traced back to the early days of development of genetic algorithms, initially as attempts to promote population diversity [24, 25, 26] in order to improve global search capabilities. However, as a side-effect, it was found that these diversity maintenance techniques could be also used to locate multiple optimal solutions. One of the earliest studies attempting to induce niching behaviour in a Genetic Algorithm (GA) is probably Cavicchios dissertation [24], where several schemes were proposed in which offspring directly replace the parents that produce them. These so called pre-selection schemes were later generalized by De Jong in a scheme called *crowding* [25]. Some years later, Goldberg and Richardson proposed *fitness sharing* [27] as alternative niching scheme, and both methods have been quite influential on later developments.

Whereas crowding attempts to restrict recombination or rather replacement to individuals that are similar, fitness sharing explicitly penalizes candidate solutions that are near to each other in terms of "similarity" and reduces their fitness accordingly. Both methods suggest to use a search space distance function as similarity measure. The overall idea is of course to enable survival of "fitter as well as different" individuals occupying different promising regions of the search space. Related niching methods developed in this time include restricted tournament selection [28] and clearing [29].

However, it has been shown that the parameters introduced by these methods (i.e., niche radius and scaling factor) are rather difficult to determine [14] without any prior knowledge of a problem. Nevertheless, the notion of a niche radius and the matching metaphorical idea of a sub-population or species has dominated the multimodal optimization field for quite some time. From the large number of proponents of niche radius based approaches, we name two examples: species conservation [5] and niching in evolution strategies with its dynamic peak identification [30].

At some point, it became clear that niche radii, even if adapted at runtime, limits the performance of multimodal optimization algorithms, simply because they cannot be expected to correctly characterize basins of attraction. Furthermore, evidence accumulated that it was unrealistic to try to employ a single population to manage a set of basins for a prolonged time [31]. At the same time, reasoning about the nature of niching and multimodal optimization led to the belief that researchers were actually try to handle a process that consisted of two distinct parts, and their interplay [32]:

- detecting and characterizing single basins of attraction, and
- performing local optimization in the most interesting of these basins.

Optimization algorithms may perform these steps explicitly or implicitly and in a continuous or alternating fashion, as summarized in [1] and [9]. Modern multimodal optimization algorithms therefore diverge in these two major dimensions:

- How are basins identified (niching method, exploration), and
- how their local optimization is steered (base algorithm, exploitation).

Next to the previously described implicit basin identification, e.g. via restricted tournament selection, and looking beyond simple niche radius based schemes, a number of interesting approaches have been suggested. In global optimization, simple clustering [2] evolved into more sophisticated specialized methods such as topographical selection [33]. The new geometrically motivated approaches consequently also use the objective value as an additional dimension. Another related method that does so is the hill-valley test of Ursem [34] that uses additional samples between suspected cluster centers. If used economically, it can be highly beneficial for multimodal optimization methods to spend a few additional evaluations, as [35] and, more recently, [36] show. Nearest-better clustering (NBC) [1] is a similar, geometrically motivated heuristic to detect basins that attempts the same goal, but without additional sample points. Wessing et al. [37] provide a thorough comparison of some of these methods and show that ensembles may improve performance.

The second important aspect of multimodal optimization algorithms is the layout of the base algorithm, how are the different optima pursued? As already described by Mahfoud, there are both sequential and parallel approaches to this [38]. In particular, the parallel methods are mostly population-based metaheuristic methods including Evolutionary Algorithms. They use different means such as e.g. communication topologies [39] between sub-populations in order to prevent loss of information that would lead to fixation (the search space shrinks to one point). New algorithms have also been developed leveraging the unique characteristics of other meta-heuristics, e.g., Particle Swarm Optimization [40] and Differential Evolution [41]. Interested readers can find detailed information on these methods in a latest survey on niching methods [8].

Sequential algorithms tend to run repeatedly, each time saving a different optimum, and they do not necessarily possess a population. However, they need to make sure that search is started at different locations. Beasley has shown that this approach can work well [42], and other algorithms as the NEA+ [1] also followed this path.

6 Niching Competition Result Analysis

This section presents a summary of the results and analysis of participating niching algorithms in the competition series on "Niching Methods for Multimodal Optimization" that took place between 2016 and 2019. These participating algorithms and presented results provide a glimpse of the latest trends in developing effective niching algorithms and the state-of-the-art in the field.

The competition series aims to provide a common platform allowing for easy and fair comparisons across different niching algorithms. This niching competition series started its first competition in 2013, at CEC'2013, based on the niching benchmark suite "Benchmark Functions for CEC'2013 Special Session and Competition on Niching Methods for Multimodal Function Optimization" [10]. The competition has been organized by the IEEE CIS Task-force on MMO[1]. It has been held at both GECCCO and CEC conferences almost every year since 2013. The competition series has been playing a key role in engaging with the niching research community, and has received numerous entries, among which several represents the state-of-the-art niching algorithms in the field [8]. The competition series has adopted widely-used performance measures, i.e., the Average Peak Ratio and Success Rate [10], for evaluating the performance of niching methods. Most importantly, the competition provides a common platform to facilitate niching researchers to evaluate their algorithms' performance on a diverse set of test functions with different characteristics with increasing complexity. Different solution accuracy levels have been adopted. Intuitively, the higher the accuracy level, the more challenging the benchmark instance becomes for locating and maintaining all the global optima. Such setup first appeared in [41], while up until the CEC 2013 competition, the community only considered arbitrary accuracy levels that might lead to inaccurate assessment of the performance of niching approaches, since a successful located solution measured by a low accuracy level (e.g., 0.1) could be still far away from the true global optimum. In contrast, higher accuracy levels require effective search capabilities from the applied algorithm to accurately locate the basin of attraction of the global optimum.

In 2016, the competition series adopted two new performance measures, to better reflect the ability and efficiency of a niching algorithm to accurately identify the set of the known global optima, i.e., how comparable is the provided set of potential solutions with the set of known global optima, and how fast the global optima are identified by the algorithm. As a result, from 2016 onwards, the competition series included the following three new scenarios to rank participating niching algorithms based on both existing and newly proposed performance measures (see Section 3 for details on how to compute these):

- **Scenario I:** This scenario adopts the CEC 2013 competition ranking procedure [10], which is based on the average Peak Ratio measure value across all benchmark instances and accuracy measures, to facilitate straightforward comparison with all previous competition entries. This measure is essentially the same as the *recall* measure commonly-used in pattern recognition and information retrieval, where *recall* is defined as the fraction of the total amount of relevant instances that were actually retrieved.
- **Scenario II:** This scenario adopts the (static) F_1 measure (as described in Eq. 2, Section 3) to take into consideration the performance characteristics of the corresponding algorithm in terms of the *recall* and *precision* of its final solution set. The ranking procedure, in this case, is performed by the average F_1 measure value across the different benchmark instances and accuracy levels. The F_1 measure is usually understood as the product of *precision* and *recall.* Here, *precision* is

[1] http://www.epitropakis.co.uk/ieee-mmo/

defined as the fraction of relevant instances among the retrieved instances. In the case of niching (assuming maximization), *precision* is the fraction of detected peaks among all the retrieved solutions. Ideally, if all sought peaks are found and only these in the submitted entry, it would result in an F_1 value of 1 (as both recall and precision would be equal to 1).

- **Scenario III:** This scenario adopts the (dynamic) F_1 measure integral ($dynF_1$ as described in Eq. 3, Section 3) over the entire run time of an algorithm, for a benchmark instance, to take into account the computational efficiency of the submitted algorithm. The integral of the F_1 measure over time (or $dynF_1$) computes the area-under-the-curve (AUC) of F_1, divided by the maximum number of function evaluations that are allowed for that specific problem instance. It is, therefore, also bounded by the value of 1. This measure takes into account the time (in terms of the number of function evaluations) at which the peaks are detected.

It is worth noting that to accurately measure the performance of the algorithms on the three scenarios, the setup includes the option for competitors to perform actions on the submitted solution set as the budget is consumed. This is an essential feature to correctly measure the behavior of algorithms on the third scenario since an algorithm should be able to adapt its decision on which solutions are characterised as potential global optima through time. As such, three actions have been defined that gives the option to add and delete solutions from the submitted solution set as well as to reset it.

The competition result analysis starts with a presentation of the final rankings of the submitted entries and some initial statistical analysis on the reported results. It then proceeds with a short analysis per scenario. Table 1 summarizes the competition results on all three scenarios for all competition entries submitted to GECCO/CEC competitions from 2016 to 2019. For each entry, the name of the algorithm (Algorithm), the mean Peak Ratio value (μ_{PR}), the mean F_1 (μ_{F_1}) and the mean $dynF_1$ (μ_{dynF_1}) are reported along with their corresponding ranking per scenario. The last two columns denote the mean rank per algorithm for all three scenarios and the final rank across all algorithms and all scenarios[2]. Figure 3 provides the boxplots which show the variations on these performance measures for each algorithm across all accuracy levels and all benchmark instances. Note that the mean value of each distribution is marked by a (black) diamond in the plots. Some of these entries are variants from their original implementation, e.g., HillVallEA19 from the originally proposed HillVallEA. It is worth pointing out that, in most cases, differences between these variants are small, and the new versions still retain the original key ideas and principles. The top three ranking algorithms are highlighted in bold. Clearly, the best-performing entries across all three scenarios are the HillVallEA19, HillVallEA, and RSCMSA17. It is interesting to note that ranks in Scenario I and Scenario II are identical for the top six entries, showing remarkable consistency. Scenario III ranks are the same only for the top two entries. Scenario III ranks for the 3rd and 4th places are the 4th and 3rd places (i.e., swapping places) according to Scenario I and II. This suggests the amount of computational effort in detecting the sought peaks starts to differ. For Scenario III, it is worth noting the performance of NEA+ and RLSIS17 algorithms that exhibit much better performance in Scenario III comparing with the first two scenarios. This is justified from the fact that both approaches incorporate strong local search algorithms (e.g., CMA-ES) which can quickly refine potential solutions, and the submitted sets of potential solutions through out the execution run is always small and includes mostly the best-estimated solutions without adding any noise.

[2] Note that, due to space limitation on graphs, the SSGA-DMRTS-DDC and SSGA-DMRTS-DDC-F entries are denoted as SSGA-D and SSGA-DF accordingly.

Algorithm	Scenario I μ_{PR}	Scenario I Rank	Scenario II μ_{F_1}	Scenario II Rank	Scenario III μ_{dynF_1}	Scenario III Rank	Mean Rank	Final Rank
ANBNWI-DE	0.8544	5	0.8872	5	0.7268	7	17/3= 5.66	5
ASCGAf	0.6000	18	0.3244	12	0.2139	13	43/3=14.33	12
CMA	0.6680	16	0.1891	15	0.2410	12	43/3=14.33	12
HillVallEA	**0.8851**	**2**	**0.9297**	**2**	**0.8689**	**2**	6 /3= 2.00	**2**
HillVallEA19	**0.8916**	**1**	**0.9335**	**1**	**0.8827**	**1**	3 /3= 1.00	**1**
IPOP	0.3522	19	0.3083	13	0.3033	11	43/3=14.33	12
NEA1	0.6088	17	0.1187	16	0.1693	14	47/3=15.66	13
NEA2	0.7458	13	0.2408	14	0.3052	10	37/3=12.33	11
NEA2+	0.8071	9	0.8548	9	0.8064	5	23/3= 7.66	7
NMMSO	0.8254	8	0.3763	11	0.1549	16	35/3=11.66	10
NMMSO2	0.8254	7	0.3763	10	0.1549	15	32/3=10.66	9
RLSIS	0.8001	10	0.8616	7	0.5903	8	25/3= 8.33	8
RLSIS17	0.7995	11	0.8553	8	0.8007	6	25/3= 8.33	8
RSCMSA	0.8548	4	0.9103	4	**0.8299**	**3**	11/3= 3.66	4
RSCMSA17	**0.8557**	**3**	**0.9106**	**3**	0.8290	4	10/3= 3.33	**3**
SDE-Ga	0.8329	6	0.8835	6	0.3764	9	21/3= 7.00	6
SSGA	0.7552	12	0.0519	17	0.0293	19	48/3=16.00	14
SSGA-D	0.6793	15	0.0268	19	0.0565	17	51/3=17.00	16
SSGA-DF	0.7380	14	0.0482	18	0.0529	18	50/3=16.66	15

Table 1 Result summary of niching competition results for entries submitted to GECCO/CEC from 2016 to 2019.

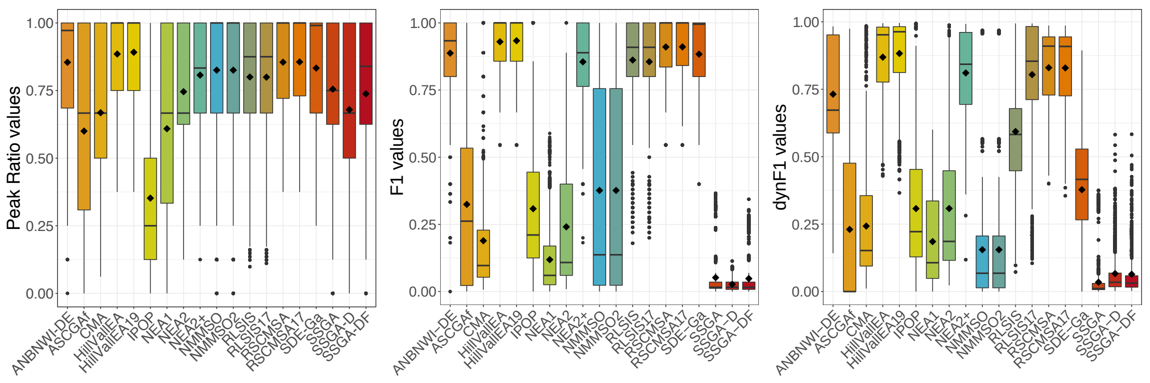

Fig. 3 Boxplots of the performance measure values across all accuracy levels and all benchmark instances. The first boxplot represents Scenario I (Peak Ratio values), the second boxplot corresponds to Scenario II (static F_1 values) and the third to Scenario III ($dynF_1$ values).

The graphical illustration of the performance distributions (in Figure 3) suggests a more clear view of the best performing algorithms across different scenarios. In Scenario I, where the peak ratio measure is used, the best performing algorithms can be clearly identified and are on par with the reported rankings in Table 1. However, rankings become a bit difficult to distinguish between the algorithms with mean and median performance values around 0.75. The performance differences of such entries become more evident in Scenario II, where the static F_1 metric considers both precision and recall of each algorithm. In this scenario, the top performers can be clearly separated from the rest ones. The top performers are ANBNWI-DE, HillVallEA, HillVallEA19, RLSIS,

RLSIS17, RSCMSA, and RSCMSA17. In Scenario III, the top-performing group of algorithms includes HillVallEA and HillVallEA19, which are followed by the RSCMSA variants (RSCMSA and RSCMSA17). The next group includes RLSIS17, NEA2+, and ANBNWI-DE.

To statistically validate the observed differences among the algorithm across the three scenarios we perform statistical analysis on the reported performance values. We first employ the Friedman rank sum test [43] to assess whether at least two algorithms exhibit significant differences in the observed performance values. The null hypothesis of the Friedman rank sum test states that the performance distributions of all samples are the same, while the alternative hypothesis states that at least two samples differ. The statistical analysis proceeds with a posthoc analysis to determine which pairs of algorithms exhibit significant differences in performance (per scenario). In this step, pairwise Wilcoxon-signed rank tests are performed on the performance samples of each pair of algorithms. In addition, to alleviate the issue of having Type I errors given multiple comparisons, the Bonferroni correction method is applied.

For all three scenarios, the applied Friedman rank-sum tests reveal statistically significant differences in performances among the submissions across all considered problem instances and levels of accuracy, with all p-values being $\ll 10^{-16}$. As such, for all scenarios, we proceed to the posthoc analysis to investigate the pair-wise performance differences among the algorithms. Figure 4 to Figure 6 present tile-plots to illustrate all pairwise differences in the observed performance samples across the different scenarios, with the outcomes of the pair-wise tests at the 5% level of significance. More specifically, the outcomes of the pairwise Wilcoxon-signed rank tests, without and with the application of the Bonferroni correction method, are provided on the left and right-hand side of the figures respectively. Each tile corresponds to a pairwise significance test between the algorithms of the corresponding row and column. The color of the tile indicates if the observed performance differences were enough to reject the null hypothesis at the 5% significance level (p-value < 0.05). As such, grey tiles indicate significant differences between the pair of algorithms, while black tiles indicate that the observed performance differences did not support the rejection of the null hypothesis, i.e., no significant differences were observed.

Both sets of significance tests suggest that the majority of the previously mentioned observations on the performance differences are statistically significant. The differences between the algorithms in the top-performing group are significant for the majority of the cases across different scenarios. There are only a few pairs of algorithms that exhibit similar behaviours, for example, NMMSO, RLSIS, and RSCMSA variants against some specific cases.

The competition series checks five different accuracy levels with complexity increases substantially as the accuracy level increases. The highest accuracy level being checked is $\epsilon = 10^{-5}$, where an algorithm (apart from identifying the regions of different global optimal solutions) has to also identify corresponding global optima with the highest accuracy. Figure 7 shows the boxplot with median and the average peak ratio (i.e., Scenario I) of all participants across all benchmark problem instances for the $\epsilon = 10^{-5}$ level of accuracy. It is noticeable that for most algorithms, their average performance is lower than their median performance. It is worth noting that the group of best-performing algorithms usually incorporate effective local search algorithms, such as CMA-ES, which enables them to accurately search and identify the global optima when approaching their basins of attraction. As a result, the top-performing algorithms can still maintain their ranks even at such a challenging accuracy level.

Similarly, for Scenario II, Figure 8 illustrates the performance distributions on precision, recall, and the F_1 measure of the algorithms across all benchmark problem instances at the highest accuracy level ($\epsilon = 10^{-5}$). It can be observed that most of the median performance values are lower

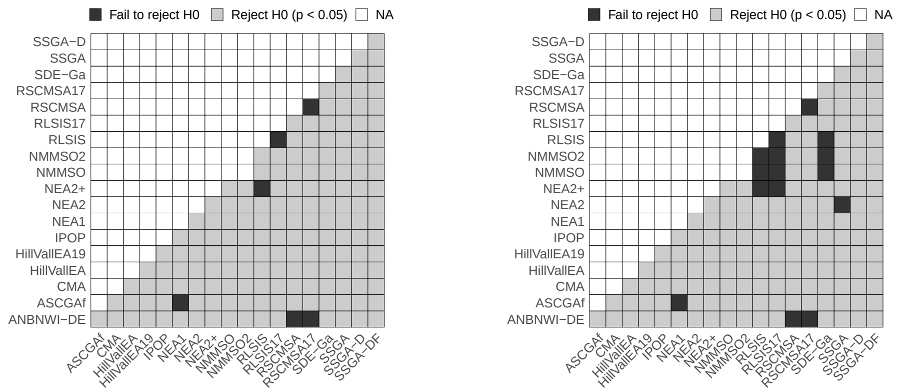

Fig. 4 Pairwise Wilcoxon statistical tests (left) with Bonferroni posthoc analysis (right) on results across all accuracy levels and all benchmark instances for Scenario I.

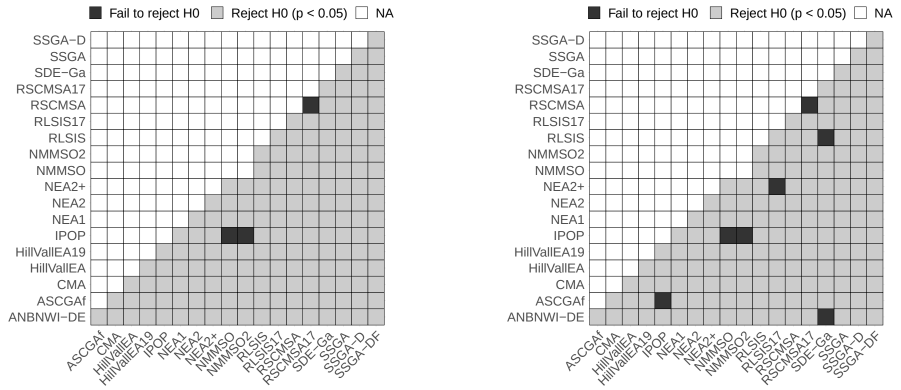

Fig. 5 Pairwise Wilcoxon statistical tests (left) with Bonferroni posthoc analysis (right) on results across all accuracy levels and all benchmark instances for Scenario II.

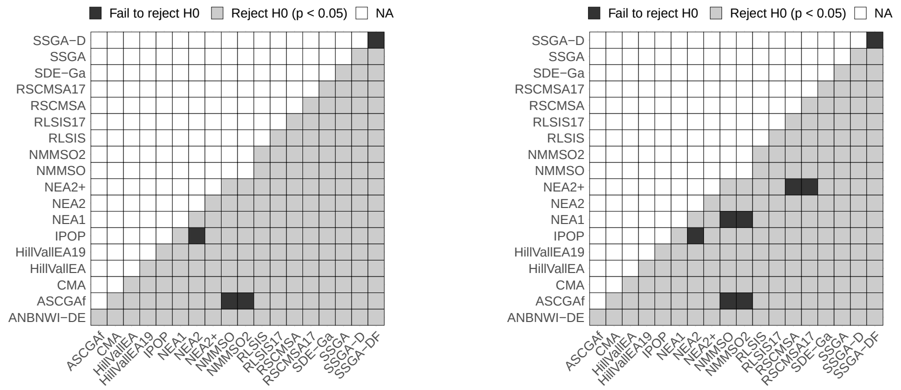

Fig. 6 Pairwise Wilcoxon statistical tests (left) with Bonferroni posthoc analysis (right) on results across all accuracy levels and all benchmark instances for Scenario III.

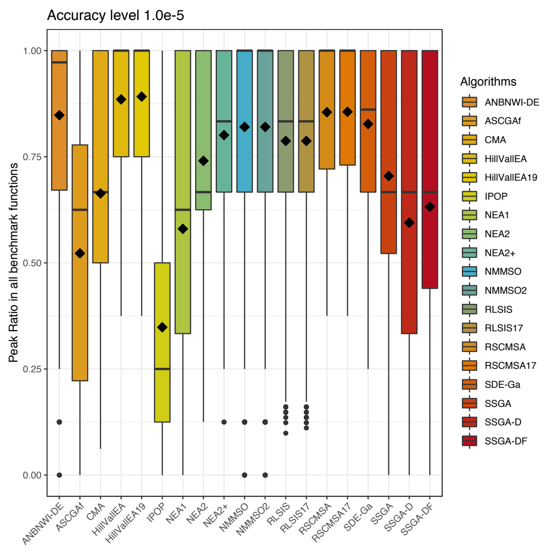

Fig. 7 Boxplot of median and the average peak ratio values for each algorithm across all benchmark instances at the accuracy level of 10^{-5}. Note that the average peak ratio value is denoted by a black diamond.

than their corresponding average values, while there is a clear separation of the top-performing algorithms from the remaining ones across all metrics. Here, high PR and SR values substantially affect the performance of an algorithm, where effective local searchers boost the top-performing algorithms substantially.

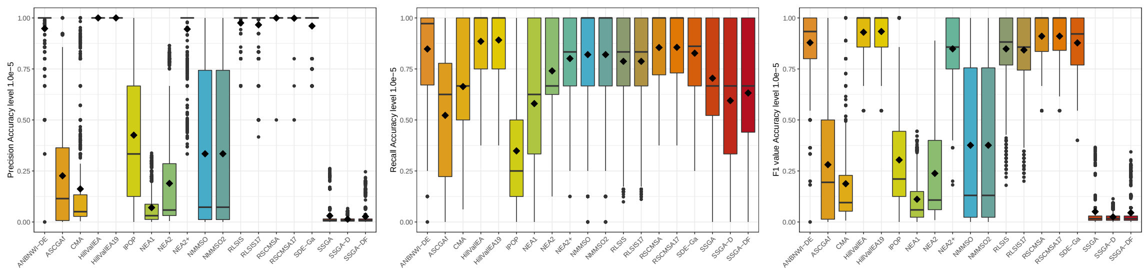

Fig. 8 Boxplots of the precision, recall and static F_1 measure for each algorithm across all benchmark instances at the accuracy level of 10^{-5}.

Figure 9 shows the performance pattern of considered algorithms across the benchmark instances for the highest accuracy level ($\epsilon = 10^{-5}$) in a heatmap plot. It is evident that the efficiency and effectiveness of the top-performing algorithms can be observed across the benchmark instances. The group of the top-performing algorithms includes HillVallEA19 and HillVallEA, RSCMSA and RSCMSA17, RLSIS and RLSIS17, the NEA+, and ANBNWI-DE, respectively, which constitute the most robust algorithms across the different scenarios.

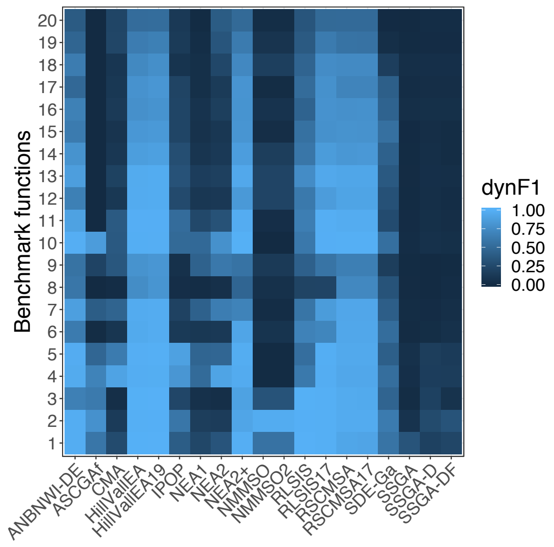

Fig. 9 Heatmap plot that demonstrates the average $dynF_1$ measure for each algorithm per benchmark instance at the accuracy level of 10^{-5}.

Here we will focus on providing some analysis on the group of top-performing algorithms. Note that the Scenario I ranking are fairly consistent with the final ranks, as shown in Table 1. The ranks are the same for the first 6 algorithms and only differ slightly after that.

The 1st and 2nd places belong to HillVallEA19 and HillVallEA respectively. The HillVallEA variants [36] employ a Hill-Valley Clustering (HVC) mechanism that combines the Hill-Valley test [34] with the notion of the nearest better tree. For the Hill-Valley test, the number of sample points to be checked is adaptively chosen. A mechanism is also in place to check if a solution belongs to the niches of its nearest better neighbours, then a new cluster is created. This process is repeated for all individuals during the selection. A restart strategy with an increased population size (starting with very small size) is adopted to allow elite individuals to continue to survive into the future iterations. The population size parameters for both the HVC and the core algorithm have been made adaptable with different problem sizes. The HillVallEA algorithm (which adopts AMaLGaM-Univariate as the core algorithm) has been shown to outperform the state-of-the-art niching algorithms of the niching competition series if more computational budget is given [36].

At the 3rd and 4th places are RSCMSA and RSCMSA17 [44]. These RSCMSA variants are developed with an effective repelling subpopulation (RS) scheme in conjunction with CMA-ES. The RS is a hybrid method built on several existing ideas such as the taboo points in Tabu search [45], normalized Mahalanobis distance [46], and the Hill-Valley function [34]. The rationale behind the choices of these techniques is to overcome the often made assumptions on distribution, size, and shape of the basins in MMO, which are impractical in many real-world problems. RSCMSA allows several subpopulations to search in parallel. The previously identified basins are marked as taboo points to avoid revisiting these basins in the future. A normalized Mahalanobis distance metric (instead of the usual Euclidean distance) is adopted to better handle non-spherical basin shape. The size of each basin is also adaptively defined by some parameters such that different sizes of basins are considered individually (instead of a uniform parameter defining all basin sizes). The Hill-Valley function is used to determine if a solution belongs to a new basin. The core algorithm adopts CMSA-ES (covariance matrix self-adaptation evolution strategy) [47]. RSCMSA builds on the strength of CMSA-ES so that it can handle ill-conditioned problems efficiently. To save computational cost, RSCMSA also employs a restart strategy with an increasing population size.

At the 7th and 8th places are NMMSO and NMMSO2 (according to Scenario I). These algorithms employ subswarms, with an exploitative particle swarm optimiser directed at each basin of attraction currently identified. Hill-Valley detection is used to distinguish these basins, and the number of swarms is unbounded. This means *a priori* information regarding the number of basins to attain is not required, but the search can be biased towards those basins of attractions that appear to be of the higher quality. New swarms are generated to look for additional basins through random seeding and splitting off exiting swarm members where Hill-Valley detection suggests they are on a different peak than the swarm they are a member of.

At the 9th place is NEA2+ (according to Scenario I), an enhanced version of NEA (Nearest-better EA) [48]. NEA builds on the idea of Nearest Better Clustering (NBC) that constructs a spanning tree to connect each solution to its nearest solution that has better fitness. NBC assumes that the best solutions are often located at different basins apart from each other and that the distances between them are usually larger than the average distance between all solutions and their nearest better neighbours. Therefore, by removing the longest edges (of the spanning tree) that are larger than the average distance between all solutions and their nearest better neighbour, clusters of solutions (or niches) can be identified. For each identified niche, NEA2+ employs CMA-ES to search its basin for an optimal solution.

7 Conclusion

In this chapter we have provided a glimpse of multimodal optimization (MMO), including its background, MMO formulation, performance measures, its historical and recent development. We also provided a result summary on the niching competition series between 2016 and 2019, hoping that such analysis can help capture the latest trends and identify techniques that have been harnessed to induce effective niching behaviours. Clearly there are still many open research questions, and we have only witnessed the early stage of development of some new breeds of niching methods.

From the previous section on niching competition results, there are clearly emerging trends in designing more powerful and robust niching algorithms, e.g., the use of more adaptive schemes for parameter specification, mechanisms to avoid revisiting the same area of the search space, restart strategies with incremental population sizes, and the adoption of individually adapted niche sizes.

Beyond developing more competent niching algorithms for just unconstrained and continuous optimization problems, e.g., as those shown in the CEC 2013 benchmark suite for MMO [10], there are abundance of research opportunities for further algorithmic development and applications of niching methods, including the following:

- Most real-world problems are highly constrained and combinatorial in nature, however, existing niching methods are mostly developed only for handling problems of unconstrained and continuous nature.
- In today's big data era, many large-scale (high dimensional) problems exist. However, most existing work on MMO have been confined to studies on low dimensional problems, and those

which are (relatively) quick to evaluate. The scalability of these methods, be it to problem size and/or expensiveness, is relatively under-explored.
- Niching methods are often associated with many user-specified parameters, which could make them difficult to use in practice. Many efforts have been attempted to make niching parameters adaptive. However, there seems to be little study on how we can make use of the preference information (which can be easily supplied by a decision maker) for niching. The benefit of this is that it will break down the barrier and allow niching methods to be more readily accepted by practitioners.
- Existing niching methods are mostly evaluated using benchmark test functions with known global optima. However, in a real-world situation, it is common that we do not have prior knowledge of the global optima (neither number nor best objective values). In such cases, how can we still measure the performance of a niching method in a meaningful manner?
- Since a decision maker may have a fatigue issue [49] (considering the cognitive load), it may be unnecessary to locate and present too many global optimal solutions. How many solutions are adequate, and how do we determine an appropriate number? There could be also some mechanism for controlling the number of solutions obtained. If a small number is chosen, then the distribution of the solutions in the set can be made more sparse (to allow a better variety of selections by the decision maker).
- Since many design problems, e.g., engineering structural design problems are typically multimodal [50], what would be the best approach to hybridize engineering optimization methods with niching methods without too much computational cost?
- Often real-world optimisation problems exhibit uncertainty or noise. There has been very little work on how optimizers developed for MMO cope on such problems.

Niching methods, which are specifically designed to locate multiple solutions in a single optimization run, are a common task that transcends the disciplinary boundaries [8]. In many real-world applications, we can see the prevalent need of discovering more than one optimal solutions, which can provide alternative solutions and additional information (of the problem under study) to a decision maker. This is in sharp contrast to the conventional view of single-optimum seeking behaviour of an optimization algorithm. As we face ever more challenging and complex problems in the big data era, there will be increasing demands on designing more powerful niching algorithms for handling ever more challening real-world MMO problems. Niching as a research topic is currently experiencing some new found rejuvenation. It is our hope that this chapter will draw more attention to the significance and benefits of doing niching, and attract even more researchers to contribute to this classic and yet fascinating area of research.

References

1. Mike Preuss. *Multimodal Optimization by Means of Evolutionary Algorithms.* Springer, 2015.
2. Aimo Törn and Antanas Žilinskas. *Global Optimization.* Springer, 1989.
3. Stephen Boyd, Stephen P Boyd, and Lieven Vandenberghe. *Convex optimization.* Cambridge university press, 2004.
4. Marco Locatelli and Fabio Schoen. *Global optimization: theory, algorithms, and applications*, volume 15. Siam, 2013.
5. Jian-Ping Li, Marton E. Balazs, Geoffrey T. Parks, and P. John Clarkson. A species conserving genetic algorithm for multimodal function optimization. *Evolutionary Computation*, 10(3):207–234, 2002.

6. Mike Preuss, Simon Wessing, Günter Rudolph, and Gabriele Sadowski. Solving phase equilibrium problems by means of avoidance-based multiobjectivization. In Janusz Kacprzyk and Witold Pedrycz, editors, *Springer Handbook of Computational Intelligence*, Springer Handbooks, pages 1159–1171. Springer, 2015.
7. Swagatam Das, Sayan Maity, Bo-Yang Qu, and Ponnuthurai Nagaratnam Suganthan. Real-parameter evolutionary multimodal optimization – a survey of the state-of-the-art. *Swarm and Evolutionary Computation*, 1(2):71–88, 2011.
8. Xiaodong Li, Michael G. Epitropakis, Kalyanmoy Deb, and Andries Petrus Engelbrecht. Seeking multiple solutions: An updated survey on niching methods and their applications. *IEEE Trans. Evol. Comput.*, 21(4):518–538, 2017.
9. Simon Wessing. *Two-stage methods for multimodal optimization.* PhD thesis, Technische Universität Dortmund, 2015.
10. Xiadong Li, Andries Engelbrecht, and Michael G. Epitropakis. Benchmark functions for CEC'2013 special session and competition on niching methods for multimodal function optimization. Technical report, RMIT University, Evolutionary Computation and Machine Learning Group, Australia, 2013.
11. Mike Preuss and Simon Wessing. Measuring multimodal optimization solution sets with a view to multiobjective techniques. In Michael Emmerich, Andre Deutz, Oliver Schütze, Thomas Bäck, Emilia Tantar, Alexandru-Adrian Tantar, Pierre Del Moral, Pierrick Legrand, Pascal Bouvry, and Carlos A. Coello, editors, *EVOLVE – A Bridge between Probability, Set Oriented Numerics, and Evolutionary Computation IV*, volume 227 of *Advances in Intelligent Systems and Computing*, pages 123–137. Springer, 2013.
12. Pascal Kerschke, Hao Wang, Mike Preuss, Christian Grimme, André H. Deutz, Heike Trautmann, and Michael T. M. Emmerich. Search dynamics on multimodal multiobjective problems. *Evol. Comput.*, 27(4):577–609, 2019.
13. Kalyanmoy Deb. *Genetic Algorithms in multimodal function optimization (Master thesis and TCGA Report No. 89002).* PhD thesis, Tuscaloosa: University of Alabama, The Clearinghouse for Genetic Algorithms, 1989.
14. David E. Goldberg, Kalyanmoy Deb, and Jeffrey Horn. Massive multimodality, deception, and genetic algorithms. In R. Männer and B. Manderick, editors, *PPSN 2*, Amsterdam, 1992. Elsevier Science Publishers, B. V.
15. Marcus Gallagher and Bo Yuan. A general-purpose tunable landscape generator. *IEEE Transactions on Evolutionary Computation*, 10(5):590–603, 2006.
16. Jani Rönkkönen, Xiaodong Li, Ville Kyrki, and Jouni Lampinen. A framework for generating tunable test functions for multimodal optimization. *Soft Computing*, 15(9):1689–1706, 2011.
17. Gulshan Singh and Kalyanmoy Deb. Comparisons of multi-modal optimization algorithms based on evolutionary algorithms. In *Proc. of the Genetic and Evolutionary Computation Conference 2006 (GECCO'06)*, pages 1305–1312, Washington, USA, 2006.
18. Bo-Yang Qu and Ponnuthurai Nagaratnam Suganthan. Novel multimodal problems and differential evolution with ensemble of restricted tournament selection. In *Evolutionary Computation (CEC), 2010 IEEE Congress on*, pages 1–7, July 2010.
19. Simon Wessing. The multiple peaks model 2. Algorithm Engineering Report TR15-2-001, Technische Universität Dortmund, 2015. `https://ls11-www.cs.uni-dortmund.de/_media/techreports/tr15-01.pdf`.
20. Mike Preuss and Christian Lasarczyk. On the importance of information speed in structured populations. In *Parallel Problem Solving from Nature - PPSN VIII*, volume 3242 of *Lecture Notes in Computer Science*, pages 91–100. Springer, 2004.
21. Ali Ahrari and Kalyanmoy Deb. A novel class of test problems for performance evaluation of niching methods. *IEEE Transactions on Evolutionary Computation*, 22(6):909–919, 2018.
22. Kalyanmoy Deb and Amit Saha. Multimodal optimization using a bi-objective evolutionary algorithm. *Evolutionary Computation*, 20(1):27–62, 2012.
23. Khulood Alyahya, Kevin Doherty, Ozgur Akman, and Jonathan Fieldsend. Robust multi-modal optimisation. In *Proceedings of the 2018 Conference on Genetic and Evolutionary Computation (GECCO'18)*, pages 1783–1790, June 2018.
24. Daniel Joseph Cavicchio. *Adapting Search Using Simulated Evolution, PhD Thesis.* PhD thesis, University of Michigan, Ann Arbor, Michigan, 1970.
25. Kenneth Alan De Jong. *An analysis of the behavior of a class of genetic adaptive systems.* PhD thesis, University of Michigan, 1975.
26. John Henry Holland. *Adaptation in Natural and Artificial Systems.* University of Michigan Press, Ann Arbor, Michigan, 1975.
27. David E. Goldberg and Jon Richardson. Genetic algorithms with sharing for multimodal function optimization. In *Proceedings of the Second International Conference on Genetic algorithms and their application*, pages 41–49. Lawrence Erlbaum Associates, Inc., 1987.

28. Georges R. Harik. Finding multimodal solutions using restricted tournament selection. In Larry Eshelman, editor, *Proc. of the Sixth International Conference on Genetic Algorithms*, pages 24–31, San Francisco, CA, 1995. Morgan Kaufmann.
29. Alain Pétrowski. A clearing procedure as a niching method for genetic algorithms. In *Proc. of the 3rd IEEE International Conference on Evolutionary Computation*, pages 798–803, 1996.
30. Ofer M. Shir. Niching in evolution strategies. In Hans-Georg Beyer, editor, *GECCO '05: Proceedings of the 2005 conference on Genetic and evolutionary computation*, pages 865–872, New York, NY, USA, 2005. ACM Press.
31. Mike Preuss, Lutz Schönemann, and Michael Emmerich. Counteracting genetic drift and disruptive recombination in (μ +/, λ)-EA on multimodal fitness landscapes. In *Proceedings of the 2005 conference on Genetic and evolutionary computation*, GECCO '05, pages 865–872. ACM, 2005.
32. Mike Preuss. Niching prospects. *Bioinspired Optimization Methods and their Applications*, pages 25–34, 2006.
33. Aimo Törn and Sami Viitanen. Topographical global optimization. In Christodoulos A. Floudas and Panos M. Pardalos, editors, *Recent Advances in Global Optimization*, Princeton Series in Computer Sciences, pages 384–398. Princeton University Press, 1992.
34. Rasmus K. Ursem. Multinational evolutionary algorithms. In Peter J. Angeline, editor, *Proceedings of the Congress of Evolutionary Computation (CEC 99)*, volume 3, pages 1633–1640. IEEE Press, 1999.
35. Catalin Stoean, Mike Preuss, Ruxandra Stoean, and Dumitru Dumitrescu. Multimodal optimization by means of a topological species conservation algorithm. *IEEE Transactions on Evolutionary Computation*, 14(6):842–864, 2010.
36. Stef C. Maree, Tanja Alderliesten, Dirk Thierens, and Peter A. N. Bosman. Real-valued evolutionary multi-modal optimization driven by hill-valley clustering. In Hernán E. Aguirre and Keiki Takadama, editors, *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2018, Kyoto, Japan, July 15-19, 2018*, pages 857–864. ACM, 2018.
37. Simon Wessing, Günter Rudolph, and Mike Preuss. Assessing basin identification methods for locating multiple optima. In *Advances in Stochastic and Deterministic Global Optimization*, pages 53–70. Springer, 2016.
38. Samir W. Mahfoud. A comparison of parallel and sequential niching methods. In *Proceedings of the 6th International Conference on Genetic Algorithms*, pages 136–143, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc.
39. Xiaodong Li. Niching without niching parameters: Particle swarm optimization using a ring topology. *IEEE Transactions on Evolutionary Computation*, 14(1):150–169, 2010.
40. Xiaodong Li. Developing niching algorithms in particle swarm optimization. In BijayaKetan Panigrahi, Yuhui Shi, and Meng-Hiot Lim, editors, *Handbook of Swarm Intelligence*, volume 8 of *Adaptation, Learning, and Optimization*, pages 67–88. Springer Berlin Heidelberg, 2011.
41. Michael G. Epitropakis, Vassilis P. Plagianakos, and Michael N. Vrahatis. Finding multiple global optima exploiting differential evolution's niching capability. In *IEEE Symposium on Differential Evolution (SDE)*, 2011.
42. David Beasley, David R. Bull, and Ralph R. Martin. A sequential niche technique for multimodal function optimization. *Evolutionary Computation*, 1(2):101–125, June 1993.
43. Myles Hollander, Douglas A. Wolfe, and Eric Chicken. *Nonparametric Statistical Methods.* Wiley, 3 edition edition, 2013.
44. Ali Ahrari, Kalyanmoy Deb, and Mike Preuss. Multimodal optimization by covariance matrix self-adaptation evolution strategy with repelling subpopulations. *Evolutionary Computation*, 25(3):439–471, 2017.
45. Fred Glover. Future paths for integer programming and links to artificial intelligence. *Computers and Operations Research*, 13(5):533–549, May 1986.
46. Ofer M. Shir, Michael Emmerich, and Thomas Bäck. Adaptive niche radii and niche shapes approaches for niching with the cma-es. *Evol. Comput.*, 18(1):97126, March 2010.
47. Hans-Georg Beyer and Bernhard Sendhoff. Covariance matrix adaptation revisited – the CMSA evolution strategy. In Günter Rudolph, Thomas Jansen, Nicola Beume, Simon Lucas, and Carlo Poloni, editors, *Parallel Problem Solving from Nature – PPSN X*, pages 123–132, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.
48. Mike Preuss. Improved topological niching for real-valued global optimization. In *Applications of Evolutionary Computation*, volume 7248 of *Lecture Notes in Computer Science*, pages 386–395. Springer, 2012.
49. Barry Schwartz. *The Paradox of Choice: Why More Is Less.* Harper Perennial, 2004.
50. Md. Jakirul Islam, Xiaodong Li, and Kalyanmoy Deb. Multimodal truss structure design using bilevel and niching based evolutionary algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '17, page 274281, New York, NY, USA, 2017. Association for Computing Machinery.