technische universität
dortmund

The Multiple Peaks Model 2

Simon Wessing

Algorithm Engineering Report
TR15-2-001
April 2015
ISSN 1864-4503

Faculty of Computer Science
Algorithm Engineering (Ls11)
44221 Dortmund / Germany
http://ls11-www.cs.uni-dortmund.de/

The Multiple Peaks Model 2

Simon Wessing

We present the *multiple peaks model 2*, a refinement of several existing test problem generators. The improvements pertain to the initialization of the problem instance, which guarantees to exactly produce a certain number of local optima, and the shape of the peaks, which are now modeled by a function that avoids the disadvantages of Gallagher's Gaussians and the function used by Preuss and Lasarczyk. Concern regarding the latter is that the worst objective value of the problem cannot be bounded easily, while the former are criticized for their steep slope. We also demonstrate how to determine the attraction basin any point in the search space belongs to.

1 Description of the Problem Generator

Many test problems in real-valued optimization do not provide information about important problem properties, as, e. g., the positions of all local optima and the corresponding attraction basins. Knowing these characteristics would be helpful in many cases of benchmarking of optimization algorithms. Since Eiben and Jelasity [1] suggested to use parametrized problem *generators* that can produce test instances randomly but with controllable difficulty, several problem formulations appeared for which these new requirements can be handled well [6, 2, 8]. The common ground of all these is that the objective function is generated by taking the maximum or minimum of several unimodal functions.

Here, we propose a refined version of this approach, which shall be named multiple peaks model 2 (MPM2), in recognition of the similarity to extensions to the generator in [6] made in [5] under the name multiple peaks model. The generator produces multimodal problem instances by combining several randomly distributed peaks. Hence, the problems are irregular and non-separable, which are also important features of difficult real-world problems. The problem is defined by the following formulas:

$$f(\boldsymbol{x}) = 1 - \max\{g(\boldsymbol{x}, \boldsymbol{p}) \mid \boldsymbol{p} \in P\} \tag{1}$$

$$g(\boldsymbol{x}, \boldsymbol{p}) = \frac{h_{\boldsymbol{p}}}{1 + \frac{\mathrm{md}(\boldsymbol{x}, \boldsymbol{p})^{s_{\boldsymbol{p}}}}{r_{\boldsymbol{p}}}} \tag{2}$$

$$\mathrm{md}(\boldsymbol{x}, \boldsymbol{p}) = \sqrt{(\boldsymbol{x} - \boldsymbol{p})^\top \boldsymbol{\Sigma}_{\boldsymbol{p}}^{-1} (\boldsymbol{x} - \boldsymbol{p})} \tag{3}$$

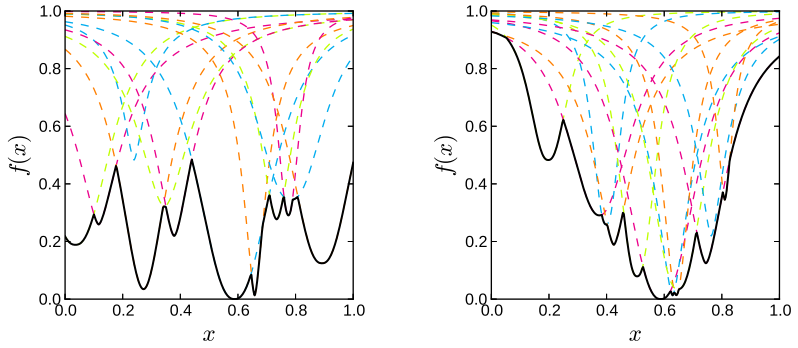

Figure 1: One-dimensional MPM2 functions (solid black) with their individual peaks (dashed). On the left, we have a random, and on the right a funnel topology. These example functions have ten optima each.

The objective function is given in (1). It takes the minimum of $N_{\text{peaks}} = |P|$ unimodal functions (2) around peak positions $\boldsymbol{p} \in P$. This has the advantage that local optima with known positions are created, which is necessary to calculate some quality indicators. The principle is illustrated in Figure 1. Rönkkönen et al. [8] criticize the exponential decay of the Gaussians used in [2]. Therefore, we are using the polynomial form in (2). Each of these functions is associated with parameters $h_{\boldsymbol{p}}$, $s_{\boldsymbol{p}}$, and $r_{\boldsymbol{p}}$ for height, shape, and radius, respectively. The idea of random shape and radius parameters is taken from [6]. By slightly deviating from locally quadratic behavior ($s_{\boldsymbol{p}} = 2$), we intend to increase the difficulty for local search algorithms. Radii $r_{\boldsymbol{p}}$ influence the size of attraction basins, and thus the probability to place a starting point in the basin. Optima with small attraction basins will be difficult to find [9, 8]. Additionally, a randomly drawn covariance matrix $\boldsymbol{\Sigma}_{\boldsymbol{p}}$ belongs to each peak. This matrix is used to create the optima's basins as rotated hyperellipsoids, by calculating the Mahalanobis distance in (3). All mentioned parameters are drawn randomly during initialization and then stored. By convention, we will always set $\max\{h_{\boldsymbol{p}} \mid \boldsymbol{p} \in P\} = 1$ and use a box-constrained search space $\mathcal{X} = [0, 1]^n$. The former has the advantage that objective function values are always in $[0, 1]$, which makes it easy to convert from maximization to minimization and to calculate some quality indicators. The latter provides some protection from numerical problems, which would appear when calculating small differences of large numbers. The matrix $\boldsymbol{\Sigma}_{\boldsymbol{p}}$ is generated in the same way as in [2], by creating a random rotation matrix $\mathbf{R}$ first, according to the method of Rudolph [7]. Then a vector $\boldsymbol{v}$ of random values $v_i \sim U(0.0025, 0.0525)$ is used to create $\boldsymbol{\Sigma}_{\boldsymbol{p}} = \mathbf{R}^\top(\boldsymbol{v}\mathbf{I})\mathbf{R}$.

Note that the number of optima ν can be smaller than N_{peaks}, because peaks can be masked by others. All previous landscape generators [6, 2, 8] suffer from this problem. We therefore employ a sophisticated initialization procedure, to obtain problems with a given number of optima. This heuristic is shown in Algorithm 1 (with a slightly overloaded notation, as $\boldsymbol{p}$ rather denotes the peak object including parameters than only the position). First, it generates a problem instance with ν peaks and iteratively reduces the radii until at least 80% of all peaks are local optima. Then, the still missing optima

Algorithm 1 Initialization of MPM2

Input: number of optima ν, number of variables n, topology
Output: test problem instance

$h_{\min} \leftarrow 0.5$; $h_{\max} \leftarrow 0.99$
$s_{\min} \leftarrow 1.5$; $s_{\max} \leftarrow 2.5$
$r_{\min} \leftarrow 0.25\sqrt{n}$; $r_{\max} \leftarrow 0.5\sqrt{n}$
$\boldsymbol{p}_1 \leftarrow$ randomUniformPeak(1, $U(s_{\min}, s_{\max})$, $U(r_{\min}, r_{\max})$) // global optimum
$P \leftarrow \{\boldsymbol{p}_1\}$
if topology is random **then**
  $P \leftarrow P \cup \{\nu - 1$ additional random uniform peaks$\}$
else if topology is funnel **then**
  $P \leftarrow P \cup \{\nu - 1$ additional randomly clustered peaks around $\boldsymbol{p}_1\}$
end if
$f \leftarrow$ createInstance(P, topology)
while |localOptima(f)| $< 0.8 \cdot |P|$ **do**
  for all $\boldsymbol{p} \in P$ **do**
    $r_{\boldsymbol{p}} \leftarrow 0.95 \cdot r_{\boldsymbol{p}}$ // reduce radii of all peaks
  end for
end while
while |localOptima(f)| $< \nu$ **do**
  $\nu_{\text{prev}} \leftarrow$ |localOptima(f)|
  while $\nu_{\text{prev}} + 1 \neq$ |localOptima(f)| **do** // do rejection sampling
    if topology is random **then**
      $\boldsymbol{p} \leftarrow$ randomUniformPeak($U(h_{\min}, h_{\max})$, $U(s_{\min}, s_{\max})$, $U(r_{\min}, r_{\max})$)
    else if topology is funnel **then**
      $\boldsymbol{p} \leftarrow$ clusteredPeak($U(h_{\min}, h_{\max})$, $U(s_{\min}, s_{\max})$, $U(r_{\min}, r_{\max})$, $\boldsymbol{p}_1$)
    end if
    $\nu_{\text{prev}} \leftarrow$ |localOptima(f)|
    $f \leftarrow$ createInstance($P \cup \{\boldsymbol{p}\}$, topology)
  end while
  $P \leftarrow P \cup \{\boldsymbol{p}\}$
end while
return f

are created by adding random peaks by rejection sampling. This means that a randomly drawn peak is only accepted if it increases the number of optima by one. Unfortunately, determining the number of optima ν takes $O(N_{\text{peaks}}^2)$ time, because we have to test for each $\boldsymbol{p} \in P$ if $f(\boldsymbol{p}) = 1 - h_{\boldsymbol{p}}$. If the condition is fulfilled, an optimum is located at $\boldsymbol{p}$. In practice, this makes the use of test problems of this kind infeasible for more than a few hundred optima, because the complete initialization procedure has cubic worst-case time complexity.

The algorithm is also capable of generating two different global structures. The first structure represents a more or less stationary case, where there is no trend in the depths of

Algorithm 2 getBasinOptimum($\boldsymbol{x}$)

Input: solution $\boldsymbol{x}$
Output: position of local optimum

1: $\boldsymbol{p}_{\text{curr}} \leftarrow \boldsymbol{x}$
2: **repeat**
3: $\quad \boldsymbol{p}_{\text{prev}} \leftarrow \boldsymbol{p}_{\text{curr}}$
4: $\quad \boldsymbol{p}_{\text{curr}} \leftarrow \arg\max\{g(\boldsymbol{p}_{\text{curr}}, \boldsymbol{p}) \mid \boldsymbol{p} \in P\}$
5: **until** $\boldsymbol{p}_{\text{prev}} = \boldsymbol{p}_{\text{curr}}$ // if fulfilled, we have arrived at local optimum
6: **return** $\boldsymbol{p}_{\text{curr}}$

local optima and locations of optima are distributed uniformly over the search space. This structure is called the random topology. The other one contains one large funnel, that is, optima are clustered around the global one and the depths are negatively correlated with distance from the global optimum. These two topologies are chosen, because it is expected that they represent two extremes on the difficulty scale of multimodal problems – at least for global optimization [10, p. 11]. This belief is reiterated in [9], where the authors speak of isolated and embedded global optima, and in [4].

Clustered peaks are drawn from a normal distribution $N(\boldsymbol{p}_1, \frac{n}{36}\mathbf{I})$, where $\mathbf{I}$ is the identity matrix. More complicated arrangements with more than one funnel are of course possible in principle. The function createInstance in Algorithm 1 takes the set of peaks and the desired topology as inputs. The topology is necessary as argument, because if we want a funnel structure, additionally to the clustering also the height values are redistributed among the peaks so that they shrink with increasing Euclidean distance from the global optimum $\boldsymbol{p}_1$.

To identify the attraction basin a point belongs to, we are using the heuristic displayed in Algorithm 2. From an arbitrary position $\boldsymbol{x} \in \mathcal{X}$, the algorithm jumps to the peak $\boldsymbol{p}_{\text{curr}}$ which is responsible for $f(\boldsymbol{x})$. As $\boldsymbol{p}_{\text{curr}}$ itself may be masked by another peak, the procedure is iterated until $\boldsymbol{p}_{\text{curr}}$ is an optimum. Thus, the set basin($\boldsymbol{x}^*$) for an optimum position $\boldsymbol{x}^*$ is approximated as

$$\text{basin}(\boldsymbol{x}^*) \approx \{\boldsymbol{x} \in \mathcal{X} \mid \text{getBasinOptimum}(\boldsymbol{x}) = \boldsymbol{x}^*\}\,.$$

Note that this realization via Algorithm 2 does not exactly emulate the behavior of a gradient descent algorithm with infinitely small step size, because we may jump over small basins that a descent algorithm would have converged to. Thus, this approach should be seen rather as an approximation. However, when assessing a set of solutions $\mathcal{P}$, the error introduced by this approach should decrease with increasing amount of local search that has been spent on $\mathcal{P}$.

2 Conclusion

The test problem generator MPM2 fulfills the same desirable properties as the max-set of Gaussians by Gallagher and Yuan [2]. They characterize appropriate test functions as

difficult to solve using simple methods such as hill climbing algorithms (P1); nonlinear, nonseparable, and nonsymmetric (P2); scalable in terms of problem dimensionality (P3); scalable in terms of time to evaluate the objective function (P4); tunable by a small number of user parameters (P5); able to be generated at random and difficult to reverse engineer (P6); and exhibiting an array of landscape-like features (P7). Property P4 is of course not really required; it can be imitated by considering different budgets of function evaluations. A corollary of P6 is that the global optimum should be distributed uniformly in $\mathcal{X}$, to avoid unintentional advantages for some solution strategies. This requirement is fulfilled by MPM2, but according to the authors not given in the black-box optimization benchmarking setup [3].

Furthermore, we have developed cost-efficient heuristics to initialize an instance with a given number of optima and to approximately determine the attraction basin a point belongs to.

References

[1] Agoston E. Eiben and Mark Jelasity. A critical note on experimental research methodology in EC. In *IEEE Congress on Evolutionary Computation (CEC)*, volume 1, pages 582–587, 2002.

[2] Marcus Gallagher and Bo Yuan. A general-purpose tunable landscape generator. *IEEE Transactions on Evolutionary Computation*, 10(5):590–603, 2006.

[3] Nikolaus Hansen, Anne Auger, Steffen Finck, and Raymond Ros. Real-parameter black-box optimization benchmarking 2010: Experimental setup. Technical Report RR-7215, INRIA, 2010.

[4] Monte Lunacek and Darrell Whitley. The dispersion metric and the CMA evolution strategy. In *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*, GECCO '06, pages 477–484. ACM, 2006.

[5] Mike Preuss. *Multimodal Optimization by Means of Evolutionary Algorithms*. Springer, 2015. (in print).

[6] Mike Preuss and Christian Lasarczyk. On the importance of information speed in structured populations. In Xin Yao, Edmund K. Burke, José A. Lozano, Jim Smith, Juan Julián Merelo-Guervós, John A. Bullinaria, Jonathan E. Rowe, Peter Tiňo, Ata Kabán, and Hans-Paul Schwefel, editors, *Parallel Problem Solving from Nature – PPSN VIII*, volume 3242 of *Lecture Notes in Computer Science*, pages 91–100. Springer, 2004.

[7] Günter Rudolph. On correlated mutations in evolution strategies. In R. Männer and B. Manderick, editors, *Parallel problem solving from nature 2*, pages 105–114. Elsevier, 1992.

[8] Jani Rönkkönen, Xiaodong Li, Ville Kyrki, and Jouni Lampinen. A generator for multimodal test functions with multiple global optima. In Xiaodong Li, Michael Kirley, Mengjie Zhang, David Green, Vic Ciesielski, Hussein Abbass, Zbigniew Michalewicz, Tim Hendtlass, Kalyanmoy Deb, KayChen Tan, Jürgen Branke, and Yuhui Shi, editors, *Simulated Evolution and Learning*, volume 5361 of *Lecture Notes in Computer Science*, pages 239–248. Springer, 2008.

[9] Aimo Törn, Montaz M. Ali, and Sami Viitanen. Stochastic global optimization: Problem classes and solution techniques. *Journal of Global Optimization*, 14(4):437–447, 1999.

[10] Aimo Törn and Antanas Žilinskas. *Global Optimization*, volume 350 of *Lecture Notes in Computer Science*. Springer, 1989.