Applied Mathematics, 2016, 7, 736-752
Published Online April 2016 in SciRes. http://www.scirp.org/journal/am
http://dx.doi.org/10.4236/am.2016.77067

Scientific Research Publishing

Cluster Search Algorithm for Finding Multiple Optima

John Guenther, Herbert K. H. Lee

Department of Applied Mathematics and Statistics, University of California, Santa Cruz, CA, USA
Email: jguenthe@soe.ucsc.edu, herbie@soe.ucsc.edu

Received 13 March 2015; accepted 25 April 2016; published 28 April 2016

Abstract

The black box functions found in computer experiments often result in multimodal optimization programs. Optimization that focuses on a single best optimum may not achieve the goal of getting the best answer for the purposes of the experiment. This paper builds upon an algorithm introduced in [1] that is successful for finding multiple optima within the input space of the objective function. Here we introduce an alternative cluster search algorithm for finding these optima, making use of clustering. The cluster search algorithm has several advantages over the earlier algorithm. It gives a forward view of the optima that are present in the input space so the user has a preview of what to expect as the optimization process continues. It employs pattern search, in many instances, closer to the minimum's location in input space, saving on simulator point computations. At termination, this algorithm does not need additional verification that a minimum is a duplicate of a previously found minimum, which also saves on simulator point computations. Finally, it finds minima that can be "hidden" by close larger minima.

Keywords

Bayesian Statistics, Treed Gaussian Process, Emulator, DBSCAN, Optimization

1. Introduction

The global minimum, which is the traditional focus of optimization, may not be as robust as another minimum, since a small change in the input may lead to a large change in the output, resulting in an inferior output. An example of this is shown in **Figure 1**. An optimum that is smoother than a global minimum with an average value over the region of interest that is lower than that of a global optimum may be preferable. For this minimum, a small change in the input does not affect the output nearly as much. **Figure 1** shows two quadratic functions with univariate input x and univariate output y. The one with a lower minimum has more curvature,

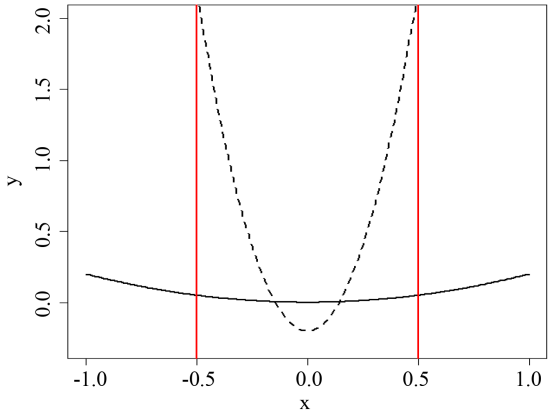

Figure 1. Robustness Illustration: The two vertical lines represent the region of interest. The two minima in the figure are placed at the origin to make visual comparison easier. The minimum with more curvature does have a lower function value. However, the smoother minimum has a lower average value in the region of interest and it remains fairly constant in value within the region while the minimum with more curvature varies much more in value in the region.

while the other is rather flat. The region of interest is between the two vertical lines. Although the function with higher curvature has the lower minimum, it rises above the other curve in the region of interest. Choosing the one with the flat curve will give the user a more stable output within the region of interest. In statistical terms, it is more robust. We want to avoid a knife's edge solution that will rapidly become suboptimal with small changes in inputs. [1] previously introduced a method for classifying the relative merits of the optima once they are found, but the search for multiple optima can be rather expensive in terms of time required to compute Black Box function values at input points. This paper proposes an algorithm to improve the efficiency and effectiveness of the search for optima. Our new cluster search algorithm not only eliminates about 25% of this search expense, but also gives the user a preview of the values and location of the minima yet to be found. Additionally, the cluster search algorithm may find minima hidden by larger minima not found by the algorithm in [1]. Without loss of generality we focus on minimization, as maximization can be obtained by minimizing the negative of the function.

Our framework is that of statistical emulation [2]. We build an efficient statistical model that approximates the unknown objective function, which may be expensive to evaluate. We use our statistical model to guide the optimization search, the accuracy of which is improved by new function evaluations guided by a sequential experimental design as the search progresses. Following the standard approach in statistical emulation [3], we use a model from the family Gaussian processes, in particular, a treed Gaussian process [4].

Section 2 reviews the topological search algorithm. Section 3 explains our new cluster search algorithm. Sections 4, 5, and 6 present illustrative examples comparing the two algorithms.

2. Topological Search Algorithm

We start with a review of the search algorithm introduced in [1]. Because it is based on surface topology it is referred to as the topological search algorithm herein. Articles on optimization that inspired the topological search algorithm are [5]-[7]. An important idea used in both [5] and [7] is that of expected improvement [8]. Using a TGP emulator to guide the search for optima is illustrated in [5] and [7]. The use of the errors and standardized errors of adaptively sampled points for verifying the emulator model was inspired by the diagnostics discussed in [6]. The approach of the topological search algorithm differs in that it uses features of the predicted surface of the emulator, rather than just using the emulator to evaluate expected improvement. In this respect, it is similar to the new cluster search algorithm presented herein.

The topological algorithm uses a hybrid approach similar to [5] by combining statistical emulation with the provably convergent direct search method, pattern search [9]. Other direct search methods could be used such as trust region optimization ([10] for example). The statistical model explores the input space for possible minimum locations which become starting points for pattern search. These locations are chosen to avoid finding minima previously explored.

The algorithm is discussed in detail in [1]. It will be summarized herein. First, the user determines a level r where $0 < r \leq 1$ relative to the global minimum, y_g, and mean simulator function value, $\overline{y}$, where these values are approximated by the emulator predicted points. The highest valued minimum of the optima search is restricted to $y_u = \left(\overline{y} - y_g\right) \cdot r + y_g$. It can be seen that low values of r target minima close to the global minimum while higher values of r target minima closer to the mean simulator value, $\overline{y}$.

Next an LHS (Latin Hypercube Sample) of input points is evaluated and the emulator uses these points to model the simulator function and predict a large number of points. The lowest valued predicted point location is sent to pattern search to find the first minimum. This becomes the approximation to the global minimum, y_g. Adaptively sampled points are added to the initial LHS to improve the emulator model and the emulator predicts another large set of predicted points. To avoid finding the same minimum, the predicted point distances from the first minimum are computed and the points at a distance d_r from the first minimum make up the input space region explored for the next minimum. This distance is determined to be $d_{\max}/2$ where $d_{\max}$ is the maximum predicted point distance. If the minimum point in this region is less than y_u, and, if the points between it and the first minimum are higher in value, this minimum point becomes the next starting point for pattern search.

At each of the next steps of the algorithm, adaptively sampled points are added to the evaluated points, the emulator models the simulator function predicting a large number of points, and distances of predicted points are computed from the minima that have been found. At some point the minimum point in the input space region explored for the next minimum is greater than y_u. The distance, d_r, is then reduced (input space region increased) until the minimum point is just equal to or less than y_u. This point becomes the next starting point for pattern search. When that starting point leads to finding a duplicate minimum, the algorithm terminates.

3. Cluster Search Algorithm

The cluster search algorithm uses of the fact that if you make a slice at a given height through a surface, and include only those points that are below the given height, the remaining points compose the part of the surface that has values lower than the given height. These points form clusters which indicate approximate locations of some minima for that surface. While a single slice at a given value gives some information about the minima of the surface, it is not sufficient to reveal all the minima. A cluster formed by this slice may include more than one minimum and may not include other minima with higher values (**Figure 2**). However, if one partitions the surface with many slices and adds the points of these partitions sequentially starting from the lowest surface values, individual clusters are formed, each of which represents a minimum. These individual clusters are differentiated by a density based clustering algorithm known as DBSCAN [11]. As these partitions are sequentially

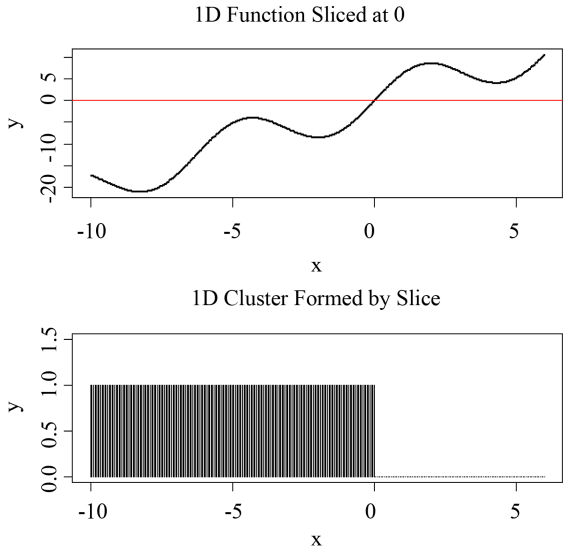

Figure 2. Example of a Cluster Formed by Single Slice of a 1D Function: The function appears in the first graph. When sliced at the height of 0, the two lowest minima make up the cluster of points less than 0. The cluster of points is illustrated in the lower graph. It includes all points with values less than 0.

added, points can fill in the spaces between clusters, but, new clusters can appear that represent other minima. Provided the surface is accurately represented by the emulator, all important minima can be found. The determination of which minima are important for the user to examine further is based on user inputs. This section explains the details of the cluster search algorithm.

In common with the topological search algorithm above, our new cluster search algorithm requires that the user decide to what level, relative to the global minimum and the mean value, the search is to be extended. Let the global minimum be y_g and the expected mean value of the objective function be approximated by the mean value $\bar{y}$ of the emulator model predicted points. We want to find local minima whose values are not too far above the global minimum. A ratio r, $0 < r \le 1$, is chosen where $r = \left(y_u - y_g\right) \Big/ \left(\bar{y} - y_g\right)$. The upper bound y_u is computed from the current observed "global" minimum and $\bar{y}$ value by $y_u = \left(\bar{y} - y_g\right) \cdot r + y_g$. While the actual value of y_g may not yet be known, the lowest value among the current minima is used for y_g. $\bar{y}$ becomes a better approximation to the actual mean value of the objective function in the input region as the minima search progresses.

For each stage of the algorithm, the treed Gaussian process emulator uses the current training point set to predict a large gridded sample of points. In the same way as the topological search algorithm, the first minimum is found by searching from the predicted point having the minimum value. This search is done by a local direct optimization routine (pattern search). Also in the same way as the topological search algorithm, as the cluster search algorithm proceeds, new function evaluations are added to the training point set via a sequential experimental design to improve the statistical model. At the first stage, we only have one identified minimum, so we use its value as the initial y_g, while in future stages, we use the smallest observed y as y_g.

After the search for the first minimum, the algorithms are much different in the way in which each determines new starting points to find the other minima in the input space. The cluster search algorithm starts each search step by partitioning the surface from the bottom to the top into sets of predicted points with approximately equally estimated function values within a given increment, $Inc = R/n_{cl}$, where $R = y_{\min}^{\text{predicted}} - y_{\max}^{\text{predicted}}$ and n_{cl} is the number of clusters. The first partition contains all points with estimated function values in the interval $\left[y_{\min}, y_{\min} + Inc\right)$. The second contains all points with estimated function values in the interval $\left[y_{\min} + Inc, y_{\min} + 2Inc\right)$. The last partition contains all points in the interval $\left[y_{\max}^{\text{predicted}} - Inc, y_{\max}^{\text{predicted}}\right]$. As each partition`s points are added in succession to a replica of the input space starting from the lowest partition, the points are either reachable to current points, or, they are unreachable based on a "minimum distance" between adjacent points. This is a known density based algorithm for determining if points belong in the same cluster (DBSCAN) which is discussed in [11]. The "minimum distance" is referred to as the ϵ neighborhood. The ϵ neighborhood is a sphere surrounding each point. Points within this distance are reachable from each other. The DBSCAN algorithm also specifies the number of points in the ϵ neighborhood, a parameter denoted $MinPts$. The parameter $MinPts$ is set to one for this application. So, if any point's ϵ neighborhood includes another point, it is a core point. (What this means with regard to the DBSCAN algorithm is that two core points whose ϵ neighborhoods include each other are in the same cluster.) In **Figure 3**, point "A" is reachable from point "B" since its ϵ neighborhood, the circle around it, includes "B". Hence, they are in the same cluster. Likewise, point "B" is reachable from point "C". So point "C" is reachable form point "A". Therefore "A", "B", "C" form a cluster. If points "A", "B", "C" were in the first cluster, the point with the lowest value would represent the first minimum. Point "N", being separated by more than the ϵ distance from any of these points would represent an approximate location for another minimum.

As more partitions are added, their points might fill in between cluster "N" and cluster "ABC" and "N" might become assimilated into a larger cluster. But, the new partition points might have a point unreachable from any of the current clusters and thus represent an approximate point for another minimum. The idea is, that as one adds the points for each partition successively, minima show up as unreachable points from the established clusters. These minima point locations along with their estimated function values are saved by the algorithm. At each search step, the lowest approximate minimum point that has not yet been searched by pattern search, becomes the next search point for pattern search.

How is this "minimum distance" ϵ determined? Think of the predicted points as a grid of equally distanced points. Along one dimension of the grid, adjacent points have a distance $w/N^{1/dim}$ from each other, where w is the input region width and N is the number of points making up the whole grid. The points along a diagonal are at a distance $\left(\sqrt{dim}\right) w/N^{1/dim}$ from each other. We would like these points to be reachable from each other. But we also would like points further than this to be unreachable. The factor of 2 was chosen. Thus, the

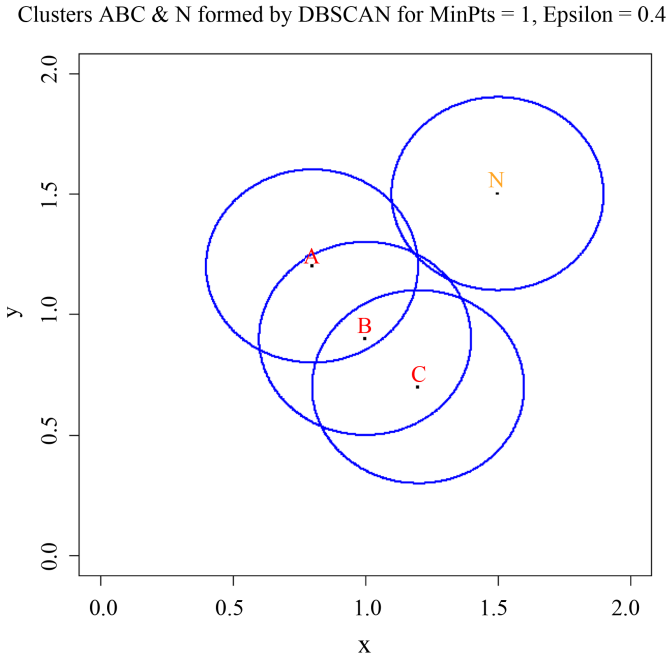

Figure 3. Example of clusters formed by DBSCAN with $MinPts = 1$ and $\epsilon = 0.4$.

minimum distance is $2\left(\sqrt{dim}\right)w/N^{1/dim}$. Minima that are closer than this are not expected unless the function is extremely variable-that is, not smooth. This is only one approach to getting the ϵ distance. It can be determined based on knowledge regarding the distance between minima. In this case, one might divide this estimated distance by some factor to get ϵ.

A specific illustration of this algorithm is given for a case where there is a test function with two minima, one being lower than the other. The negative of the test function is shown with minima represented as peaks for better visualization. Its surface is shown in **Figure 4**. It is functionally given as:

$$f(x_1, x_2) = -N_{x_1}\left(1/4, 0.1^2\right) \times N_{x_2}\left(1/2, 0.1^2\right) - 0.7N_{x_1}\left(3/4, 0.1^2\right) \times N_{x_2}\left(1/2, 0.1^2\right) \quad \text{where} \quad \left[0 \le x_1, x_2 \le 1\right]$$

The surface is divided into 25 partitions. The first four nonempty partitions are plotted in **Figure 5** (Each partition may or may not contain points, although, usually an empty partition is unusual for a dense set of points). The circles represent the ϵ neighborhood of each point with a radius of the minimum distance. Within partitions, points are ordered by ascending estimated function value. The first point in the first partition, appearing on the left hand side of the graph, has the lowest value and becomes the first minimum search point. It has an associated bold red circle at the minimum distance. Points reachable from this point are circled by lighter red circles. In the fourth nonempty partition, the third of the four points in that partition is not reachable from the points accumulated so far. It appears as a bold red circle on the right side of the graph and becomes the next estimated minimum point. The last point in this partition is reachable from this last estimated minimum point and has a light red circle at minimum distance. These two minimum points in the first four partitions represent the two estimates for the minima in the above function.

In this illustration the actual function value is used to evaluate the points. In a computer experiment, the function value is estimated by the emulator given a set of sampled points, usually starting with an LHS. The predicted points have the estimated function values that are used to create the partitions. Just as in the case of the topological search algorithm, a relatively close matching of the simulator surface is required for obtaining all the minima within the desired range. The treed Gaussian process usually provides such a matching provided enough function points have been evaluated in regions where the function is variable.

As this algorithm proceeds, the user is informed, both by a graphical display and by a print out, of the list of predicted minima that have been found and perspective minima yet to be found. The lowest valued minimum yet

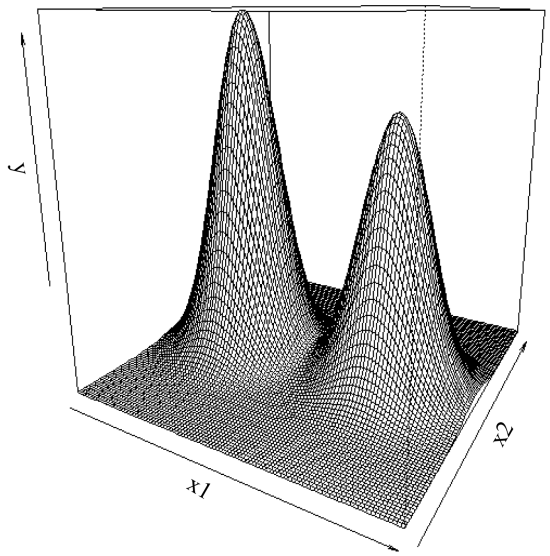

Figure 4. Perspective plot of illustration function with two minima shown as peaks.

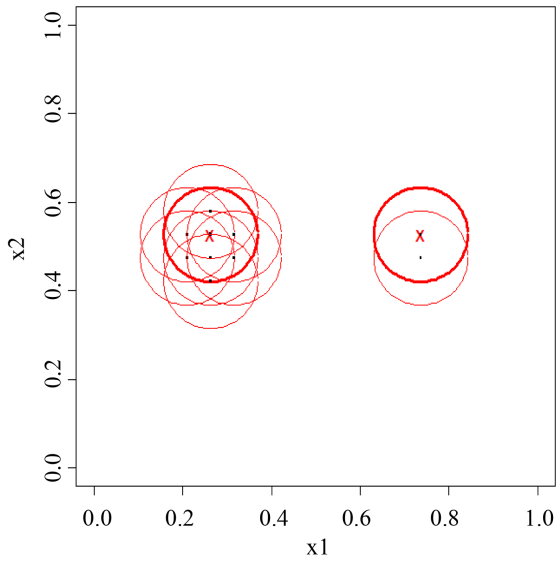

Figure 5. Illustration of clustering for the two minimum function.

to be found in this list becomes the next search point for pattern search. When there are no more minima in the list to be found, the search is ended and the utility of the minima is estimated as in [1]. This is an improvement over the previous topological search algorithm in that pattern search does not need to verify the search has ended by searching for a duplicate minimum. Another improvement here is that the user is informed of the progress of the optima search in so far as how many optima to expect and how much more processing is needed to complete the search. The topological algorithm only informs the user of what optima have been found.

The step size is determined differently than it is for the topological search algorithm. Although the starting step size used in pattern search starts at the step size used by the topological search algorithm, the step size is less than that of the topological search algorithm for subsequent steps since the cluster search algorithm, on average, locates the search point closer to the actual minimum. This results in fewer pattern search point evaluations.

An overview of the algorithm is given by the following pseudo code. The algorithm starts by evaluating an LHS of training points, the size of which varies depending on the variability of the simulator function, then it initializes the base *B* and the level parameter *r*, the loop count (ct_{loop}) of the main loop, the number of partitions for the input space, the predicted grid size, the tolerance for pattern search, the minimum distance between minima, the number of loop passes between optima search steps (inc_{loop}), the number of adaptively sampled points for each loop execution, and, optionally, the error and/or standard deviation limits at which to begin optima search steps. Then it proceeds as follows:

Pseudo Code for Optimization

```
Begin loop for minimum search:
    Increment ct_loop
    Model simulator function with emulator and current training points and predict a large grid of points
    If modulo(ct_loop/inc_loop) = 0 and first optima search step
        Invoke pattern search at minimum predicted point x_min to find first minimum
    Elseif modulo(ct_loop/inc_loop) = 0 and not first search step
        Invoke cluster search algorithm to
            Partition the predicted surface points from bottom to top
            Successively apply DBSCAN to each partition to determine minimum points
            Order minimum points by ascending minimum value
            Print minima locations, values, and whether found or not found for user
            If all minima found, terminate processing
            Else invoke pattern search at the location of the lowest minimum value not found
    Endif
    Adaptively sample points for model improvement
End loop for minimum search
```

End Pseudo Code for Optimization

4. First Illustrative Example

In this example, the cluster search algorithm and the topological search algorithm are compared for the modified Schubert test function. They are compared both as to their ability to find the minima specified by the user and the number of point evaluations needed to find the minima. It will also be shown that the user can anticipate the results far more readily when using the cluster search algorithm. The function, taken from [1], is given in equation below:

$$\begin{aligned} f\left(x_1,x_2\right) = &\left[\sum_{j=1}^{5} 0.9\,j\cos\left(\left(j+1\right)\left(x_1+0.25\right)+j\right)\right]\times\left[\sum_{j=1}^{5} 0.9\,j\cos\left(\left(j+1\right)\left(x_2+0.25\right)+j\right)\right] \\ &\times\exp\left(-\left(x_1-1\right)^2-\left(x_2-1\right)^2\right)+0.25\exp\left(-800\left(\left(x_1-1.2\right)^2+\left(x_2-0.68\right)^2\right)\right) \\ &+0.15\exp\left(-\left(x_1-0.68\right)^2-\left(x_2-1.2\right)^2\right)I\left(\sqrt{\left(x_1-0.68\right)^2+\left(x_2-1.2\right)^2}<0.1\right) \end{aligned}$$

where $0\le x_i\le 2$ for $i\in\{1,2\}$ and $I(\cdot)$ is the indicator function equal to one when its argument is true and zero otherwise.

The differences from the original Schubert function [12] are described in detail in [1]. The essential difference is that the variables are restricted to the interval $[0,2]$, rather than the original interval $[-10,10]$. The perspective plot is shown in **Figure 6**. The minima of the function are listed in **Table 1**.

For both the cluster search algorithm run and the topological search algorithm run, the run parameters were as follows:

- The value of $r=0.4$, so the minima specified were within $y_g < y_{\min} \le \bar{y}$ such that $\dfrac{y_{\min}-y_g}{\bar{y}-y_g}\le 0.4$.

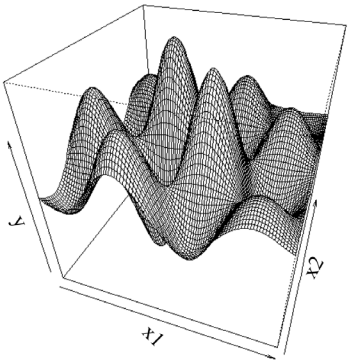

Figure 6. Modified schubert test function for first illustration of optimization search comparison: The perspective plot shown is the negative of the modified Schubert function since the negative shows the minima as peaks for better visualization. The input space contains eight minima in the domains of the variables x_1 and x_2 which vary from 0 to 2.

Table 1. Modified schubert function minima: This table has the eight minima locations given in columns x_1 and x_2 and values given in column $f(x_1, x_2)$.

x_1	x_2	$f(x_1, x_2)$
1.20	0.68	−9.684
0.68	1.20	−9.584
0.17	0.68	−6.225
0.68	0.17	−6.225
1.20	1.72	−4.446
1.72	1.20	−4.446
0.17	1.72	−2.934
1.72	0.17	−2.934

- Initially, 100 points were sampled using the R function “improvedLHS” from the R package “lhs” [13].
- For each successive step four training points are selected by the sequential experimental design based on their standard deviations (large standard deviations given preference) and distance from previous training points. This is done to improve the statistical model as processing progresses.
- The predicted point size for each search step is 2000. The cluster search algorithm makes use of gridded points, whereas the topological search algorithm typically uses a LHS sample.
- A minimum search was conducted at each step. For other examples that are compared, minima searches occur at different step intervals so that additional training points can be added to improve the statistical model.

Each illustrative example (this being the first) is compared in these ways: 1) The minimum found; 2) The number of training points computed to find the minima; 3) The look ahead data provide by the cluster search algorithm (not available for the topological search algorithm).

4.1. Minima Found

The minimum found by both algorithms are in **Table 2**. Both algorithms found the same minima which are at the required level given the ratio $r = 0.4$. Recall $y_u = y_g + r(\bar{y} - y_g)$ where $y_g = -9.687$ and $\bar{y} \approx 0.0098$, yielding $y_u \approx -5.808$.

4.2. Number of Training Points

The number of training points computed by the cluster search algorithm was 313 as compared with the number

computed by the topological search algorithm which was 405. The percent difference is 29%. Insight into this difference can be deduced from **Figure 7**. The black points that cover the input region that are not near the found minima are the training points added to improve the statistical model. The black points concentrated at the found minima are computed by pattern search. Notice that these points are concentrated near the minima for the cluster search algorithm while these points are more scattered for the topological search algorithm. The reason for this is that the cluster search algorithm starts its search, on average, closer to the minima. This lessens the number of search points computed by pattern search. Furthermore, the concentration of red points in the graph for the topological search algorithm are due to the final pattern search that finds a duplicate minimum. This final search is not required by the cluster search algorithm and is another reason why fewer points are computed by the cluster search algorithm. This difference in the number of computed points is significant for a computer experiment when the Black Box function takes significant time to return a computed point.

4.3. Look ahead Capability of the Cluster Search Algorithm for the First Illustrative Example

The cluster search algorithm has the capability to "look ahead" and show the user possible minima that will be included in the output. The user may then decide if he/she wishes to continue the search for minima or be satisfied with those currently found. There are two outputs that support this capability: A graphical output and a printed output which are explained below. The topological search algorithm has no such capability. It is "blind" in regards to what minima might exist and what their potential locations and values are.

The graph in **Figure 8** shows the location of the detected minima in the perspective of the two influential variables. The four red "X's are the estimated locations of the minima as found by the cluster search algorithm

Table 2. Minima found for algorithm runs: Four minima were found by both algorithm runs. The data were very similar for each algorithm. Minor differences are due to the randomness in each algorithm. The cutoff for the minima found is $y_u \approx -5.808$. Since the next lowest minimum is -4.446 each algorithm found minima at the desired level. The table has locations given in columns x_1 and x_2 and values given in column $f(x_1, x_2)$.

x_1	x_2	$f(x_1, x_2)$
1.202	0.682	−9.687
0.684	1.205	−9.590
0.164	0.683	−6.229
0.683	0.165	−6.229

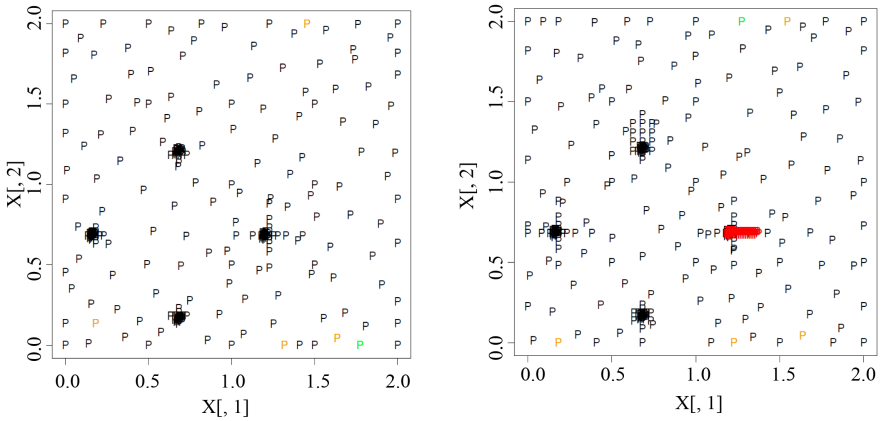

Figure 7. Training point comparison for example 1: The graph on the left shows the training points computed by the cluster search algorithm. The graph on the right shows the training points computed by the topological search algorithm. The black points are the training points. The red points in the topological search graph are the pattern search points which find a duplicate minimum. The four orange and one green point are the added points by the last search step to improve the statistical model.

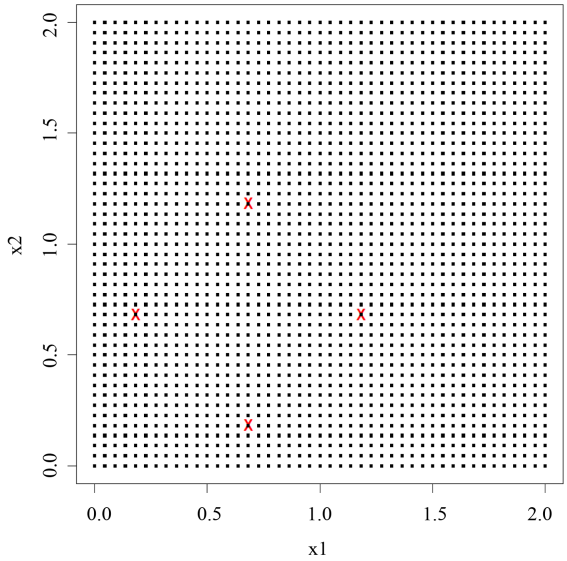

Figure 8. Minima detected by cluster search algorithm in Step 2: The minima detected are shown as the red "X's. The gridded predicted point locations are shown as the black square dots. In search Step 2, there are four estimated minima detected coinciding with the number of minima found by the run.

that meet the requirements of the user. The black square dots are the gridded prediction point locations, the values of which are used by the cluster search algorithm to estimate the minima locations.

Regarding the printed output, the cluster search algorithm outputs the estimated number of minima that are compatible with the user's input requirements and lists their estimated locations and values. It identifies those that have been already found by a distance computation (a small distance (≤ 0.05) from a known minimum indicates a minimum has already been found) and those that have not been found. Its next minimum search starts at the estimated location of the minimum with the lowest estimated value which has not been found. In this case, for search step 2, the next search is for the second minimum in the list at location $\left(x[1]=0.6818, x[2]=1.1818\right)$ with minimum value $y=-9.4249$. This information is contained in **Table 3**. This information along with the graph comprises a look ahead for the user.

5. Second Illustrative Example

In this example, the cluster search algorithm and the topological search algorithm are compared for a function with six distributed minima shaped like Gaussian curves which are relatively close together. In this case both algorithms find the six minima. The function is given in equation below:

$$\begin{aligned} f(x_1,x_2) = &\exp\left(-1/(0.1)^2 * \left((x_1-1/4)^2+(x_2-1/4)^2\right)\right) \\ &+\exp\left(-1/(0.1)^2 * \left((x_1-1/2)^2+(x_2-1/4)^2\right)\right) \\ &+\exp\left(-1/(0.1)^2 * \left((x_1-3/4)^2+(x_2-1/4)^2\right)\right) \\ &+\exp\left(-1/(0.1)^2 * \left((x_1-1/4)^2+(x_2-1/2)^2\right)\right) \\ &+\exp\left(-1/(0.1)^2 * \left((x_1-1/2)^2+(x_2-1/2)^2\right)\right) \\ &+\exp\left(-1/(0.1)^2 * \left((x_1-3/4)^2+(x_2-1/2)^2\right)\right) \end{aligned}$$

where $0 \leq x_i \leq 2$ for $i \in \{1,2\}$. The perspective plot of the function is shown in **Figure 9**. The capability of the algorithms to resolve close minima is tested by this function. The minima are given in **Table 4**.

For both the cluster search algorithm run and the topological search algorithm run, the run parameters were as

Table 3. Estimated minima locations and values for search Step 2: The table shows the estimated minimum locations and values along with their distances from the known minimum and the indicator of whether they have been found.

x_1	x_2	$f(x_1, x_2)$	Distance	Found (Yes/No)
1.1818	0.6818	−9.5315	0.02	Yes
0.6818	1.1818	−9.4249	0.72	No
0.1818	0.6818	−6.1589	1.02	No
0.6818	0.1818	−6.1505	0.72	No

Table 4. Six close normal minima test function: The six minima in this table have locations given in columns x_1 and x_2 and values given in column $f(x_1, x_2)$. Each minimum is similar in value. The test is intended to check whether the minimum at (0.50, 0.25) which is interior to the others is resolved by the two algorithms.

x_1	x_2	$f(x_1, x_2)$
0.25	0.25	−1.004
0.50	0.25	−1.006
0.75	0.25	−1.004
0.25	0.50	−1.004
0.5,	0.50	−1.006
0.75	0.50	−1.004

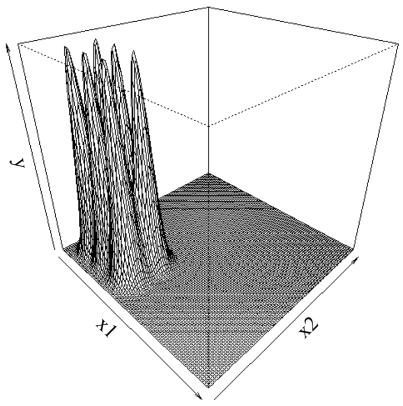

Figure 9. Six close normal minima test function for optimization search comparison: The perspective plot shown is the negative of the function. As with the modified Schubert function, the negative shows the minima as peaks for better visualization. There are six close minima in the input space of the important variables x_1 and x_2.

follows:

- The value of $r = 0.4$, so the minima specified were within $y_g < y_{\min} \le \bar{y}$ such that $\frac{y_{\min} - y_g}{\bar{y} - y_g} \le 0.4$.

  Because the minima are close together, allowing for variation in the minimum values is more likely to resolve some minima that are not defined as well by the statistical model.
- Initially, 150 points were sampled using the R function “improved LHS” from the R package “lhs”.
- For each successive Step 10 training points are selected by a sequential experimental design based on their standard deviations (large standard deviations given preference) and distance from previous training points. The closeness of the minima were expected to test the statistical model accuracy, so more points were needed to make the statistical model match the function.
- The predicted point size for each search step is 2000. The cluster search algorithm makes use of gridded

points, whereas the topological search algorithm typically uses a LHS sample.

- A minimum search was conducted every other step in order to have more improvement is the statistical model for search steps.

5.1. Minima Found

The minima found by both algorithms are in **Table 5**. Both algorithms found the same minima which are above the required level given the ratio $r = 0.4$. In this case, $y_g = -1.006$ and $\bar{y} \approx 0.045$, yielding $y_u \approx -0.621$, which did not actually come into play since all minima are approximately around −1.00. The data were very similar for each algorithm. Minor differences are due to the randomness in each algorithm.

5.2. Number of Training Points

The number of training points computed by the cluster search algorithm was 566 as compared with the number computed by the topological search algorithm which was 716. The percent difference is 27%. Insight into this difference can be deduced from **Figure 10**. The black points that cover the input region that are not near the found minima are the training points added to improve the statistical model. The points concentrated at the found minima are computed by pattern search. Notice that these points are concentrated near the minima for the cluster search algorithm while these points are more scattered for the topological search algorithm. This is evident for the topological search algorithm by the "tails" which move toward the minima. This means more points are computed by pattern search for the topological search algorithm since the search starts, on average, further from the minima. This difference as well as the final search by the topological search algorithm (the red

Table 5. Minima found for algorithm runs: Six minima were found by both algorithm runs. The table has locations given in columns x_1 and x_2 and values given in column $f(x_1, x_2)$.

x_1	x_2	$f(x_1, x_2)$
0.749	0.499	−1.004
0.500	0.500	−1.006
0.500	0.250	−1.006
0.250	0.500	−1.004
0.750	0.249	−1.004
0.251	0.251	−1.004

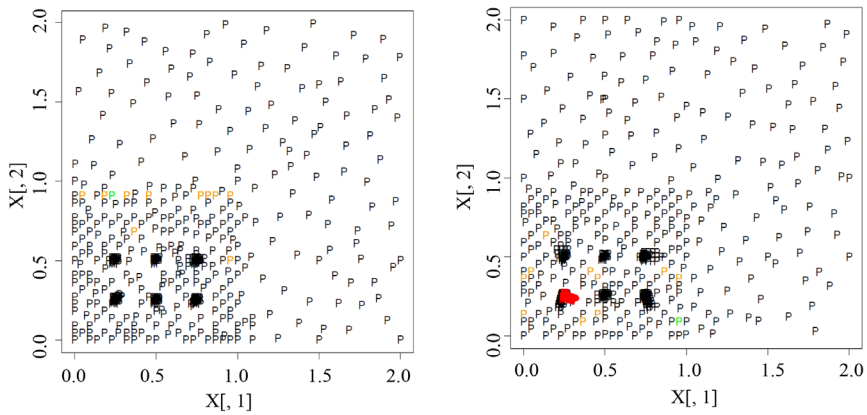

Figure 10. Training point comparison for example 1: The graph on the left shows the training points computed by the cluster search algorithm. The graph on the right shows the training points computed by the topological search algorithm. The black points are the training points. The red points in the topological search graph are the pattern search points which find a duplicate minimum. The ten orange points and one green point are the added points by the last search step to improve the statistical model.

points) are the reason for the increase in the number of training points computed by the topological search algorithm. This increase is significant for a computer experiment when the Black Box function takes significant time to return a computed point.

5.3. Look ahead Capability of the Cluster Search Algorithm for the Second Illustrative Example

This example also shows that the cluster search algorithm is dependent on how well the statistical model matches the function surface. In the third search step, only three minima were detected. In the fourth, fifth, and final search steps, the cluster search algorithm detected all six minima. This is a result of the fact that, as training points are added via the sequential experimental design, the statistical model is a better match to the function surface. This is more evident as the function surface is more variable. The program has the capability to suspend search steps until the errors in added training points show, on average, values below a user specified limit. However, in the cases presented herein, the accuracy of the statistical model is managed by the number of training points added between search steps. The two graphs in **Figure 11** are for search Steps 3 and 4. The one for search Step 3 shows 3 minima are detected, the graph for Step 4 shows that all six minima are detected. In each graph, the red X's are the estimated locations of the minima as found by the cluster search algorithm that meet the requirements of the user. The black square dots are the gridded prediction point locations, the values of which are used by the cluster search algorithm to estimate the minima locations.

The printed output for the second illustrative example for search Step 4 is shown in **Table 6**.

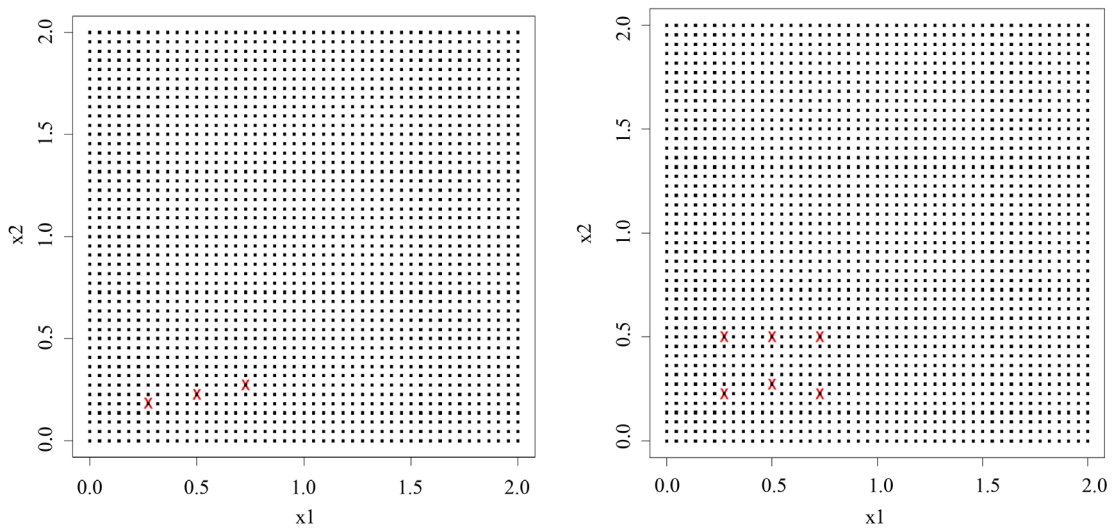

Figure 11. Minima detected by cluster search algorithm in Step 3 and Step 4: The minima detected are shown as the red X's. The gridded predicted point locations are shown as the black square dots. There are three estimated minima detected in Steps 3 and 6 estimated minima detected in Step 4.

Table 6. Estimated minima locations and values for search Step 4: The table shows the estimated minimum locations and values along with their distances from the known minimum and the indicator of whether they have been found.

x_1	x_2	$f(x_1, x_2)$	Distance	Found(Yes/No)
0.5	0.5	−1.0058	0.0	Yes
0.5	0.2727	−0.9591	0.02	Yes
0.7273	0.5	−0.9572	0.02	Yes
0.2273	0.5	−0.9521	0.27	No
0.7273	0.2273	−0.9078	0.23	No
0.2727	0.2273	−0.9078	0.23	No

The information conveyed here is that six minima have been detected by the cluster search algorithm. Their locations and minimum value are estimated. The first one is within 0 units distance from a found minimum and is therefore associated with a minimum already found by the program. Minima 2 and 3 are within 0.02 units of found minima and are associated with minima already found. Minima 4, 5, and 6 are a significant distance away from found minima and therefore still need to be located by pattern search. Search Step 4 starts at the location $\left(x[1]=0.2273, x[2]=0.50\right)$ of the fourth detected minimum which has an estimated value of $y=-0.9521$. This is the lowest valued minimum to be found.

6. Third Illustrative Example

In this example, the cluster search algorithm and the topological search algorithm are compared for a function with one global skewed minimum covering a large area of the input space and a smaller normally distributed minimum on the edge of the input space. This example shows that the cluster search algorithm can detect the smaller minimum although the topological search algorithm does not find it. This is another advantage of the cluster search algorithm. The function given in equation below:

$$f\left(x_1, x_2\right) = -10\left(x_1 - 0.35\right)^{0.02} \exp\left(-\left(x_1 - 0.35\right)\right) N_{x_2}\left(1, 0.02^2\right) - N_{x_1}\left(0.1, 0.12^2\right) N_{x_2}\left(1, 0.1^2\right)$$

where $0 \le x_i \le 2$ for $i \in \{1, 2\}$. The perspective plot of the function is shown in **Figure 12**. The capability of the algorithms to resolve a minimum in the vicinity of a larger minimum is tested by this function. The minima are given in **Table 7**.

For both the cluster search algorithm run and the topological search algorithm run, the run parameters were as follows:

- The value of $r = 0.4$, so the minima specified were within $y_g < y_{\min} \le \bar{y}$ such that $\dfrac{y_{\min} - y_g}{\bar{y} - y_g} \le 0.4$. As in the example for the six close minima, allowing for variation in the minimum values is more likely to resolve some minima that are not defined as well by the statistical model.

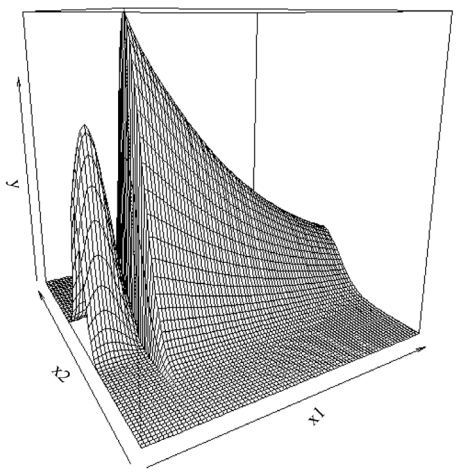

Figure 12. The perspective plot shows the negative of the function so that the minimum are peaks which can be readily seen. There is one large skewed minimum and one smaller although possibly significant normally distributed minimum.

Table 7. Skewed minimum with smaller normal minimum: The two minima in this table for this test function have locations given in columns x_1 and x_2 and values given in column $f\left(x_1, x_2\right)$.

x_1	x_2	$f\left(x_1, x_2\right)$
0.36	1.00	−19.483
0.10	1.00	−13.263

- Initially, 150 points were sampled using the R function “improved LHS” from the R package “lhs”.
- For each successive processing step, 20 training points are selected based on their standard deviations (large standard deviations given preference) and distance from previous training points. The closeness of the minima and the variability of the function are expected to test the statistical model accuracy, so more points are needed to make the surface match the function.
- The predicted point size for each search step is 2000. The cluster search algorithm makes use of gridded points, whereas the topological search algorithm typically uses a LHS sample.
- A minimum search is conducted every third processing step in order to have more improvement in the statistical model for search steps. This function has a very steep decrease near the large skewed minimum as can be seen in the perspective which shows this as a sharp increase. It is nearly discontinuous along in the vicinity of line of $x_2 = 0.35$.

6.1. Minima Found

The minima found by both algorithms are in **Table 8**. Both algorithms found the larger skewed minimum but only the cluster search algorithm found the smaller minimum. The mechanism of the topological search algorithm is to search for minima at a distance from known minima so as not to find previously found minima. It adjusts this distance if there are no minimum less than y_u in the search region until the search region includes point(s) at least a small as y_u. If this point is closer to a known minimum than one not yet found, as in this example, the known minimum is found again. When a duplicate minimum (one already found) is found, the search terminates. The smaller minimum is above the required level given the ratio $r = 0.4$. In this case, $y_g = -15.581$ and $\bar{y} \approx -1.760$, yielding $y_u \approx -10.229$.

6.2. Number of Training Points

The number of training points computed by the cluster search algorithm was 443 as compared with the number computed by the topological search algorithm which was 416. This comparison is not meaningful since the topological search algorithm did not find all the qualified minima. Insight as to why the topological search algorithm failed to find the smaller minimum can obtained by seeing the final training points for both search algorithms shown in **Figure 13**. The black points that cover the input region that are not near the found minima are the training points added to improve the statistical model. The points concentrated at the found minima are computed by pattern search. The cluster search algorithm training points show that the minima are well located for each pattern search. This is evident from the clusters of pattern search points being close to each minimum. The topological search algorithm training points would show that the large skewed minimum is located initially by a search indicated by black points. However, this previous search is obscured by a stream of red points that find the skewed minimum again, starting from a location well away from the skewed minimum. This location is on the edge of the search region which is adjusted until the edge contains points as low as or just lower than y_u. For this example, the smaller minimum is not in the search region. A default parameter was used for the search distance. Adjusting this parameter could help the topological search algorithm find the smaller minimum. However, no such adjustment is required for the cluster search algorithm.

Table 8. Minima found for algorithm runs: The two minima were found by the cluster search algorithm. The topological search algorithm found only the larger skewed minimum. The location and value for the large skewed minimum is accurately found by both algorithms. However, only the cluster search algorithm found the smaller normally distributed minimum. For the topological search algorithm, this smaller minimum is “in the shadow of” the large skewed minimum. The cutoff for the minima found is $y_u \approx -10.229$ showing that the smaller minimum, $f(x_1, x_2) = -13.263$ is in the range of qualified minima. The table has locations given in columns x_1 and x_2 and values given in column $f(x_1, x_2)$. The two right columns of the table indicates which algorithms found a given minimum.

x_1	x_2	$f(x_1, x_2)$	cluster search algorithm	topological search algorithm
0.357	0.999	−15.851	yes	yes
0.100	1.000	−13.263	yes	no

6.3. Look ahead Capability of the Cluster Search Algorithm for the Third Illustrative Example

This example shows that the cluster search algorithm detected the two minima in search Step 2. This indicates that the training points yield a statistical model that is reasonably accurate. The experimental sequential design added training points near the most variable input space of the test function. The graph in **Figure 14** shows the approximate locations of the two minima. In the graph, the two red X's are the estimated locations of the minima as detected by the cluster search algorithm that meet the requirements of the user. The black square dots are the gridded prediction point locations, the values of which are used by the cluster search algorithm to estimate the minima locations.

The printed output for the third illustrative example for search Step 2 is shown in **Table 9**.

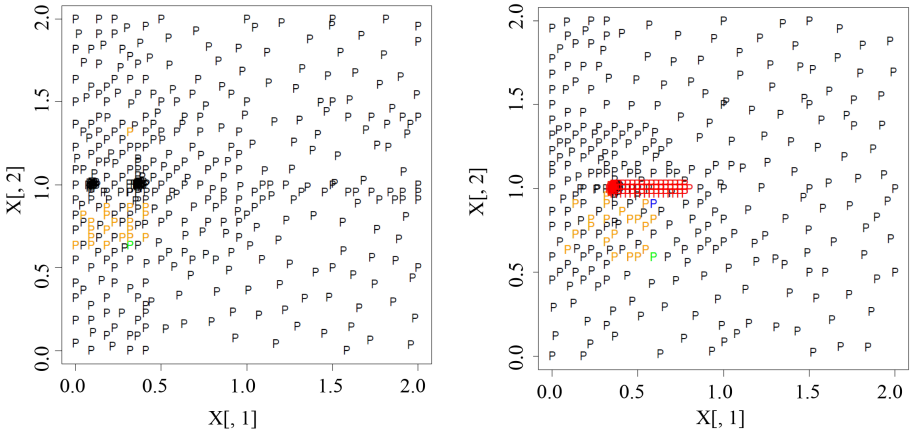

Figure 13. Training point comparison for example 3: The graph on the left shows the training points computed by the cluster search algorithm. The graph on the right shows the training points computed by the topological search algorithm. The black points are the training points. The red points for the topological search graph are the pattern search points that find the skewed minimum again, obscuring the black points of the first search for the skewed minimum. The twenty orange points and one green point (and one blue point in the topological search graph) in each graph are the added points by the last search step to improve the statistical model.

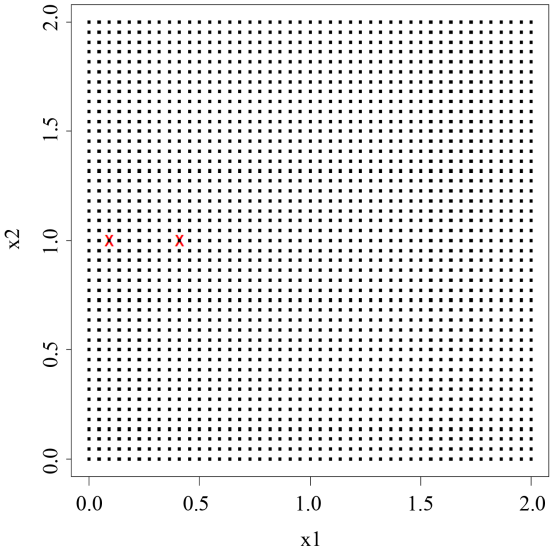

Figure 14. Minima detected by cluster search algorithm in search Step 2: The minima detected are shown as the red X's. The gridded predicted point locations are shown as the black square dots. There are two estimated minima detected in search Step 2.

Table 9. Estimated minima locations and values for search Step 2: The table shows the estimated minimum locations and values along with their distances from the known minimum and the indicator of whether they have been found.

x_1	x_2	$f(x_1, x_2)$	Distance	Found (Yes/No)
0.4091	1.0	−15.08	0.05	Yes
0.0909	1.0	−11.8545	0.27	No

The information conveyed here is that two minima have been detected by the cluster search algorithm. Their locations and minimum value are estimated. The first one is within 0.05 units distance from a found minimum and is therefore associated with a minimum already found by the program. Minimum 2 is within .27 units of a found minimum and therefore still needs to be located by pattern search. Search Step 2 starts at the location $(x[1] = 0.0909, x[2] = 1.0)$ of the second detected minimum which has an estimated value of $y = -11.8545$. This is the lowest valued minimum to be found.

7. Conclusion

We propose a new cluster search algorithm for efficiently exploring the whole input space, which is a significant improvement over earlier multi-optimum search algorithms such as that in [1]. This new algorithm has the advantage of finding the same search minima in fewer Black Box function evaluations than the one in [1], it gives the user a "look ahead" capability to foresee the outcome of the experiment, and it can find minima that are potentially hidden by larger minima which are not detected by the algorithm in [1]. It can be used to search for multiple optima and then those optima can be evaluated with the utility function of [1] for optimum selection that makes use of Bayesian decision theory to quantize the attributes of interest in optimum selection, the optimum's smoothness and value, and takes into account the user's specific needs.

References

[1] Guenther, J., Lee, H.K. and Gray, G. (2014) Finding and Choosing among Multiple Optima. *Journal of Applied Mathematics*, **5**, 300-317.

[2] Kleijnen, P.C.D. (2015) Design and Analysis of Simulation Experiments. 2nd Edition, Springer, New York.

[3] Santner, T., Williams, B. and Notz, W. (2003) The Design and Analysis of Computer Experiments. Springer, New York.

[4] Gramacy, R. and Lee, H. (2008) Bayesian Treed Gaussian Process Models with Application to Computer Modeling. *Journal of the American Statistical Association*, **103**, 1119-1130.

[5] Taddy, M., Lee, H., Gray, G. and Griffin, J. (2009) Bayesian Guided Pattern Search for Robust Local Optimization. *Technometrics*, **51**, 389-401.

[6] Bastos, L. and O'Hagan, A. (2009) Diagnostics for Gaussian Process Emulators. *Technometrics*, **51**, 425-438.

[7] Lee, H., Gramacy, R., Linkletter, C. and Gray, G. (2011) Optimization Subject to Hidden Constraints via Statistical Emulation. *Pacific Journal of Optimization*, **7**, 467-478.

[8] Jones, D., Shonlau, M. and Welch, W. (1998) Efficient Global Optimization of Expensive Black-Box Functions. *Journal of Global Optimization*, **13**, 455-492.

[9] Gray, G. and Kolda, T.G. (2006) Algorithm 856: APPSPACK 4.0: Asynchronous Parallel Pattern Search. *ACM Transactions on Mathematical Software*, **32**, 485-507.

[10] Wild, S.M. and Shoemaker, C.A. (2013) Global Convergence of Radial Basis Function Trust-Region Algorithms for Derivative Free Optimization. *SIAM Review*, **55**, 349-371.

[11] Han, J., Kamber, M. and Pei, J. (2012) Data Minig Concepts and Techniques. 3rd Edition, Elsevier Inc., Watham, 371-373.

[12] Hedar, A.R. and Fukushima, M. (2006) Tabu Search Directed by Direct Search Methods for Nonlinear Global Optimization. *European Journal of Operational Research*, **127**, 329-349.

[13] Carnell, R. (2012) Package "lhs": Latin Hypercube Samples, Version 0.10. CRAN Repository.