Journal of Advances in Applied Mathematics, Vol. 2, No. 1, January 2017
https://dx.doi.org/10.22606/jaam.2017.21001

A Robust Algorithm for Optimum Utility

John Guenther and Herbert Lee

Division of Applied Mathematics and Statistics, University of California, Santa Cruz, United States
Email: jguenthe@soe.ucsc.edu

Abstract The black box functions found in computer experiments often have multiple optima. When there are multiple optima, the goal of a computer experiment should be to determine the best optimum for the purposes of the experiment. This brings up the concept of the utility of an optimum: The degree to which the optimum fulfills the purposes of the experiment. The utility of an optima is based on measures of the variability of the surface in the tolerance region (region of interest around the optima) defined by the accuracy with which influential variables can be specified. This tolerance region can be symmetric or asymmetric. For symmetric optima the tolerance region that provides maximum utility is centered at the optimum point. For asymmetric optima, it may be advantageous to displace the center of the tolerance region from the optimum point. For example, a very skewed optimum (minimum in this case) can increase significantly in one direction but only slightly in the opposite direction from the minimum point. For such an optimum the center point of the tolerance region that provides maximum utility for the optimum is displaced from the optimum point along the direction that only slightly increases. The algorithm discussed in this paper locates the center point for the tolerance region to achieve the best utility for any optimum, symmetric or asymmetric. Making use of the surface predictions of the emulator around the minimum point, it employs pattern search to find the best center point for any optimum and for any dimensionality.

Keywords: Bayesian statistics, treed gaussian process, emulator, decision theory, optimization.

1 Introduction

The goal of optimization is to find the "best" optimum for the purposes of the computer experiment. Often there are several local optima to choose from. To determine which is best among them, the user must examine their characteristics. For example, spiky optima may not be suitable, even if they give a lower absolute value, since a small change in an input variable may result in a large change in the output. Thus it is important to be able to accurately assess the utility of different local optima. Our previous work has focused more on location of optima [1,2], and assessment of utility under the assumption that the values of the objective function are spherically symmetric in the local region around an optimum. Here we provide a more robust algorithm for evaluating utility, in that it works well for both symmetric and non-symmetric optima. This algorithm can be used for any black box function, such as a computer experiment, with a continuous output function. Difficult optimization problems are typically multimodal, and this method is widely applicable for assessing which optimum is best for the purposes of the experiment.

The utility or desirability of optima has been studied by other investigators. In [3], the utility was determined by the mean and variance where the variance was determined from a finite set of known probability distributions. In [4], the best candidate optimum was chosen based on the mean of the simulator values at some specified distance from the optimum. In [5], the solution (optimum chosen) is based on the expected value and the associated confidence interval. Although these ways of choosing optima do have value, they do not consider all aspects of the optimum that the user might find important.

A related concept is that of sensitivity analysis, where the effects of each input can be measured either locally or globally [6]. By looking at the local effect around an optimum, one could see how much change to expect, and thus how robust the optimum is, or how wide the associated confidence interval would be. There are various approaches to selecting points, using surrogate models, and computing sensitivity [7,8].

Continuing with the explanation of the algorithm presented herein, without loss of generality, we consider minimization herein, as maximization is achieved by minimizing the negative of the function. Consider the region of interest around the optimum which we herein call the tolerance region. This region is defined by the accuracy with which the influential variable's can be specified. [1] defines four measures of interest for a local minimum: 1) The minimum value (lower bound), 2) the mean value in the tolerance region, 3) the maximum value in the tolerance region (upper bound), and, 4) the range of values (upper bound minus lower bound) in the tolerance region.

Figure 1 illustrates the different measures discussed above. There are four symmetric minima each one having one "best" measure. The one in the upper left has the least range. So the output y in the tolerance region (in this case from -0.2 to 0.2) varies less than it does in the tolerance regions of the other minima (optima). The minimum in upper right has the least minimum value, so it might be preferred if the user just wants to get the optimal value in an application. The minimum in the lower left might be preferred if the user wants to be sure the output y is always less than the least maximum value. The minimum on the lower right could be preferred if the user wants to have, on average, a lower functional value. The most flexible way for a user to choose the "best" minimum is to weight his/her choice for

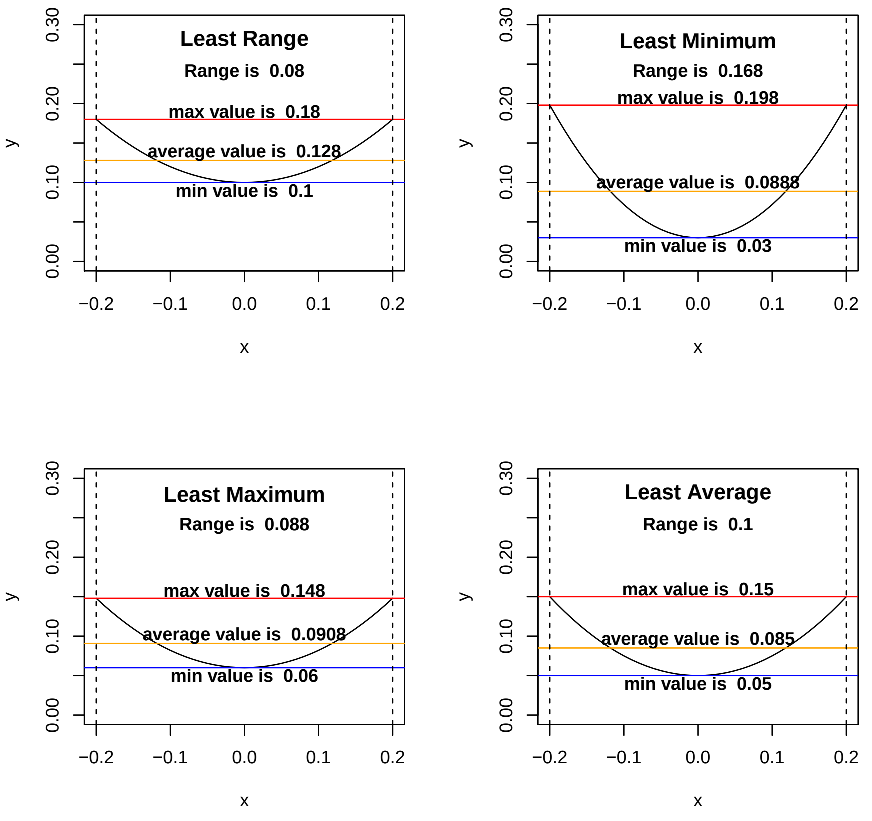

Figure 1. Utility Measures Illustration: The optimum in the upper left has the least range. The optimum in the upper right has the least minimum. The optimum in the lower left has the least maximum. The optimum in the lower right has the least average functional value.

each measure. So he/she chooses the weights for each of the four measures based on their importance to his/her application. Let the weights be $w = (w_1, w_2, w_3, w_4)$ where the weights are chosen so that they add to 1. The measures are then scaled from 0% to 100% by letting G be the lowest minimum value(global minimum), by letting B be a reference value larger than any minimum's maximum value, and by computing the scaled measures as follows:

$$m_1 = abs((y_{min} - B/(G - B)) \cdot 100\%. \tag{1}$$

$$m_2 = abs((y_{max} - B)/(G - B)) \cdot 100\%. \tag{2}$$

$$m_3 = abs((y_{average} - B)/(G - B)) \cdot 100\%. \tag{3}$$

$$m_4 = 100\% - (m_1 - m_2). \tag{4}$$

Then the utility for the minimum becomes:

$$utility = w_1 \cdot m_1 + w_2 \cdot m_2 + w_3 \cdot m_3 + w_4 \cdot m_4. \tag{5}$$

Each of the scaled measures, computed in this way, has a value from 0% to 100%. One aspect of this scaling is to move to a maximum utility framework by flipping the axis so that the more desirable values are larger (lower function output values have higher utility). Also, each minimum is related to the same base values, G and B, so their utilities can be compared. The measure for range is based on the two measures, m_1 and m_2. Notice that the smaller the range is, the larger range meaure m_4 is. Since the weights are chosen to add to 1, the utility cannot exceed 100%.

To demonstrate how this works for the minima in Figure 1, the global minimum is set to 0.03, the least minimum value in the group of four minima. It is understood that the minima, although all shown as located at $x = 0$, are at different locations in the input space. Further, the value of B will be set to 1, which is larger than any maximum value for the minima. Using weights of $w = (.25, .25, .25, .25)$ the utilities are computed and shown in the graph in Figure 2 below. The utility is marked by a red 'X' for each minimum and its value is given in a text message to the left. Minimum 4 has the greatest utility. In viewing the graph one can see that this minima, although not the best in all measures, compares favorably in all measures to the other minima.

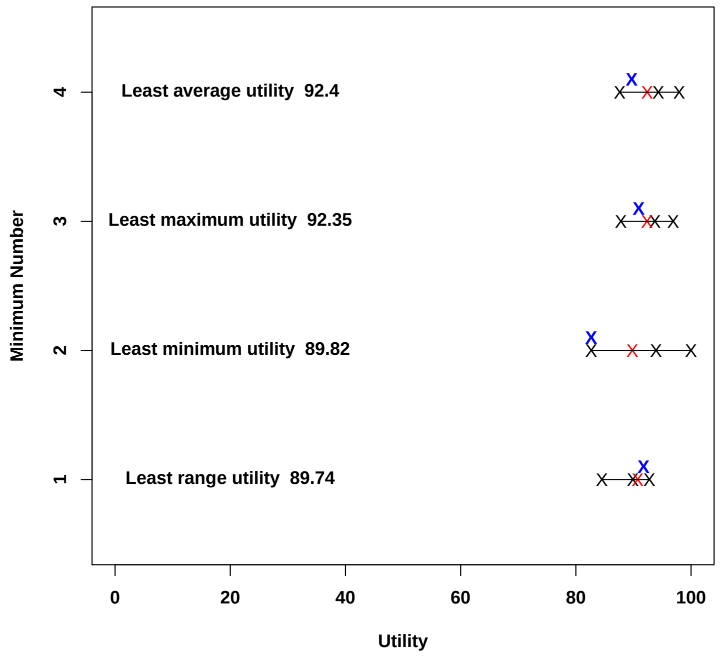

Figure 2. Utility Illustration: The utilities for the optima of Figure 1 are shown in this graph. They are numbered on the vertical axis. The three black 'X's from left to right are the measures for the maximum value, the average value, and the minimum value. The measures are switched so lesser values have more utility than larger values. The blue 'X' represents the range measure. It is offset so that its shows up better in the graph. The red 'X' is the utility obtained from the four measures.

This shows how optimum utilities are computed in [1] in the case where the measures are known. However, in the case where the minima are modeled by an emulator, the true values of the measures in the tolerance region are estimated by the emulator's predictions of the surface in the tolerance region. The uncertainty in predicting the function output values is captured by the posterior predictive distribution [9], which gives a probability distribution of the possible output values for each unobserved input. Combining the posterior predictive with a Bayesian decision theory discussed in [10] allows us to incorporate the uncertainty in the function outputs into our optimal decision. A utility function which quantifies the importance/weight of the measures needs to be chosen. One maximizes this utility function to obtain the optimal decision. If there were exact values for each measure, we would have a utility function that is a vector inner product $U(w, m^{(j)}) = w \cdot m^{(j)}$, where $w = (w_1, \ldots, w_4)$ are the weights attached to each measure by the user and $m^{(j)} = (m_1^{(j)}, \ldots, m_4^{(j)})$ are the scaled versions of the measures for a given optimum as described above. Here $j = 1 : n$ designates one of n optima. Since the $m^{(j)}$ are unknowns associated with a probability distribution function, the Bayesian decision approach employs an expected value for determining the utility: $a_j = \int_m U(w, m^{(j)}) p(m^{(j)}) dm^{(j)}$. The optimum with the maximum expected value (highest utility) is the "best" optimum. The utility values, a_j, of all of the optima rank their utility relative to each other.

Filling in the details of the utility based approach, the Treed Gaussian Process emulator predicts a representative number of surface points within the tolerance region. (The standard statistical model for emulation is the Gaussian process, a computationally accessible nonparametric model that can accommodate a range of response functions [11]. An improvement over the traditional Gaussian process is the treed Gaussian process (TGP), which has the capability of modeling functions with regions of different variability (nonstationarity) better than a stationary Gaussian process [12]. We use TGP as our emulator throughout this paper.) Fitting the emulator in a Bayesian framework allows for full accounting of uncertainty. There are several iterations for each predicted point forming a posterior probability distribution for that point. For each iteration of the surface points, a minimum, maximum, and an average are determined. These are then used to compute the scaled measures for that iteration. A Monte Carlo approximation is then made from this collection of scaled measures for a given optimum: $a_j == \int_m U(w, m^{(j)}) p(m^{(j)}) dm^{(j)} \approx \sum_{i=1}^{N} w \cdot m_i^{(j)} / N$, where N is the number of iterations. One note on the selection of B. A reasonable choice for B is the average value of all training points or the average value of the predicted values for a large random sample covering the whole input space. Values of B that are closer to the maximum y_{max} of the minimums, $max(y_{max}^{(j)})$, accentuate the differences in the optima more, although their relative utilities remain the same. Once chosen, B must be held constant for each optimum's Monte Carlo approximation to fairly compare their utilities.

Up till now, we have been discussing the computation of the utility of symmetric optima (minima). This summarizes the method for choosing optima as presented in [1]. Here there is no requirement for center of the tolerance region to be moved from the optimum point. However, not all optima are symmetric. In fact, most optima have a degree of asymmetry. The motivation for the algorithm explained herein is illustrated in Figure 3. One can see that by positioning the center of the tolerance region so that it covers the optimum point on the left boundary with the right boundary on the gently sloping side of the skewed optimum, the utility is increased substantially. Any setting of the variable x within this tolerance region achieves a relatively high output value y. The section that follows explains the algorithm for both symmetric and asymmetric optimal in detail.

2 A Robust Algorithm for Optimum Utility

The first step of the algorithm is as follows: For each optimum in the search for optima for which the utility is being determined, the emulator predicts a grid of points which is centered at the optimum point that is a cube with sides three times the length of the tolerance region. (In this discussion, for simplicity, the tolerance region is considered to be a cube such that the influential variables all have the same accuracy specified. If the variables have different accuracies, the model could be transformed so that they do have the same accuracy in the transformed units.) This grid, as a minimum, has 15 predicted points along each side with a total of 15^d points where d is the dimensionality of the input space for the influential variables. For each optimum, all the iterations for each grid point are saved.

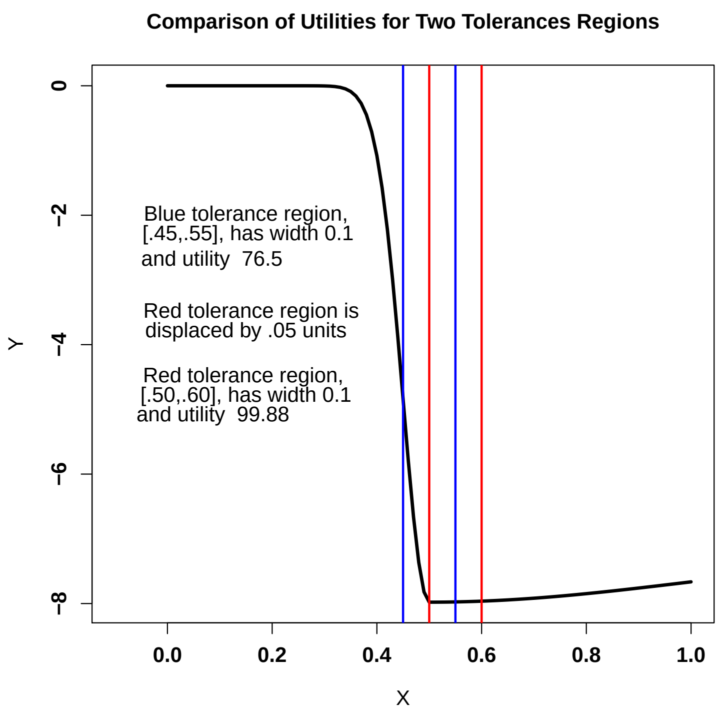

Figure 3. Comparison of Two Tolerance Regions: The graph of the skewed optimum (minimum) shows the utilities of two tolerance regions. The one with blue line boundaries for the tolerance region centered at the optimum point has a utility of 76.5%. The second one with red line boundaries for the tolerance region which includes the optimum point as its left boundary has a utility of 99.88%. The difference is due to the fact that the range within the red line tolerance boundaries is less and the maximum value is less.

The second step occurs after all important optima as determined by the investigator have been found. This step requires a utility function that assesses the utility of any location within the grid of predicted points for a given optimum as the center of a tolerance region. The utility function first determines the set of predicted points in this tolerance region. Then, for each iteration of this set of points, the minimum, maximum and average of the output values are computed. (This is similar to the procedure discussed in the previous section for symmetric optima.) The utility for this particular tolerance region is given by the Monte Carlo estimation:

$$a_j = \int_m U(w, m^{(j)}) p(m^{(j)}) \mathrm{d}m^{(j)} \approx \sum_{i=1}^{N} w \cdot m_i^j / N. \tag{6}$$

where N is the number of iterations. Because this function basically integrates a continuous function over a region, it is a continuous function for each grid point over the region of the grid of points. (Here the assumption is that the simulator function is continuous.) Therefore, a search method like pattern search can make use of this function to search for a maximum of the function in this region. The basic pattern search algorithm is explained fully in [13]. However, for this case, it need not be run asynchronously. Starting at the optimum point, pattern search can step to the point with maximum utility. Pattern search needs some initial parameters for a successful run. The step size should be at least about twice the size of the distance between two grid points along a variable's axis. This assures that it explores locations with different utilities. If the initial step size is smaller than the grid distance, it may not find the maximum. Another parameter is the ending step size. This is not as critical and can be set to one fifth the grid distance.

To illustrate this algorithm for a test function without the emulator, it is applied to the asymmetric test function. The perspective view of the function is shown in the Figure 4. Mathematically, this function is:

$$\begin{aligned} f(x_1, x_2) &= 20 \cdot x_1 + 20 \cdot x_2,\ for\ [x_1 < 0, x_2 < 0], \\ &= x_1 + 20 \cdot x_2,\ for\ [x_1 \geq 0, x_2 < 0], \\ &= 20 \cdot x_1 + 0.1 \cdot x_2,\ for\ [x_1 < 0, x_2 \geq 0], \\ &= x_1 + 0.1 \cdot x_2,\ for\ [x_1 \geq 0, x_2 \geq 0]. \end{aligned} \tag{7}$$

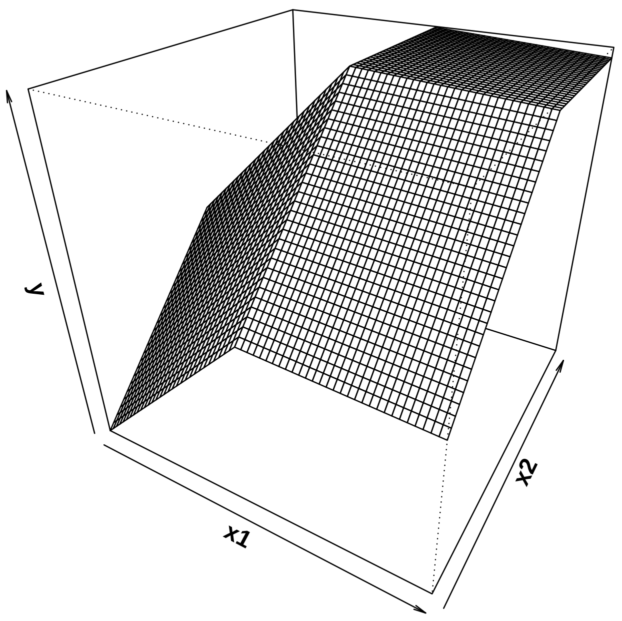

Figure 4. Asymmetric Test Function: The perspective shows the negative of the function so the minimum appears as a peak for easier visualization. The minimum occurs at the point (0,0).

The asymmetric test function, as a minimum, increases steeply as x_1 or x_2 decrease from 0. The test function only gradually increases as x_1 and x_2 both increase from 0. In the quadrant where both variables are positive, increases in x_2 do not increase the function as much as increases in x_1. The minimum has the value of 0 at (0,0). For this function the tolerance region is hypothesized as 0.05 units by 0.05 units. To determine the maximum utility for this minimum, a square grid of points of width 0.15 units by 0.15 units is calculated centered at (0,0) (three times the width of the tolerance region). Pattern search is run starting at (0,0) accessing the utility function which computes the utility of the tolerance region of a given center point. A pseudo code for this function is below. In this pseudo code, *size* is the half length of the tolerance region side and $x = (x_1, x_2)$ is the point at which the utility is computed.

```
function utility($x = (x_1, x_2), w = (w_1, w_2, w_3, w_4), G, B, size$)
    find each $y_i$ in tolerance region $[x_1 - size, x_1 + size], [x_2 - size, x_2 + size]$
    let these $y_i$ be the set $y$
    $y_{min} = \min(y)$
    $y_{max} = \max(y)$
    $y_{average} = \text{mean}(y)$
    $m_1 = \text{abs}((y_{\text{min}} - B)/(G - B)) \cdot 100\%$
    $m_2 = \text{abs}((y_{\text{max}} - B)/(G - B)) \cdot 100\%$
    $m_3 = \text{abs}((y_{\text{average}} - B)/(G - B)) \cdot 100\%$
    $m_4 = 100\% - (m_1 - m_2$
    utility $= w_1 \cdot m_1 + w_2 \cdot m_2 + w_3 \cdot m_3 + w_4 \cdot m_4$
return(utility)
```

The values for the weights are set to 0.25 for this illustration. The value for G is 0, the minimum of the function. B is set to the reference value 6 which is greater than the value of y_{max}. The half length of the tolerance region side is 0.025. (Note: The utility function is readily generalized for greater dimensions. The point becomes $x = (x_1, x_2, ..., x_d)$ and the points in the tolerance region is defined by $([x_1 - size, x_1 + size], [x_2 - size, x_2 + size], ..., [x_d - size, x_d + size])$.

A schematic of how pattern search uses this function to find the maximum utility is shown below for the simple case of one dimension. The utility function is represented as a skewed function with a maximum at $x = 0$. In Figure 5, the dotted line is the path of the utility function. The starting step size is 0.02. Pattern search starts by computing the utility at the minimum point $x = 0$ shown as the blue 'X' and then computes the utilities at 0.0 ± 0.02 shown in red as '1' and '2'. It moves to '2' at $x = 0.02$ as the new base location since the utility at $x = 0.02$ is greatest. At $x = 0.02$, it computes the utilities at 0.02 ± 0.02. (The utility at $x = 0.0$ need not be recomputed if pattern search saves values at points already visited.) Since point '3' at $x = 0.04$ has the greatest utility, pattern search moves to '3' as its next base location. This process continues in this manner for 6 steps until the base location becomes $x = 0.1$. This is the point of maximum utility. The utilities at 0.1 ± 0.02 are both lower than the utility at $x = 0.01$. At this point, pattern search reduces the step size to 0.01 and computes the utilities at 0.1 ± 0.01. Again the new points have lower values for utility. Since this is the ending step size, pattern search terminates and returns the utility at $x = 0.1$ and its location $x = 0.1$.

Continuing with the illustrative example, pattern search does 42 utility evaluations and 9 processing steps to determine the maximum utility and its location. The data for these steps is in Table 1 below. The increment between the points along a variable's axis is 0.005. The starting step size for pattern search is four times 0.005 or .02. It is increased from twice the increment 0.005 to speed up the convergence. The ending step size is 0.0025. In the final processing steps, several points could achieve the maximum utility. This is because of the discrete nature of the the utility function. Points within the grid increment share the same grid points for their utility. One could average these points' coordinates to give the center of these maximum utility points, although any one of these points could serve as a center for the tolerance region since they are very close together and have the same utility. For this data, the tolerance center point and its utility are $(x_1, x_2) = (.0475, 0475)$ with utility 98.91%. The utility as a function of the grid points is shown in Figure 6. It also shows the starting and ending points for pattern search.

A schematic for the algorithm steps is shown in Figure 7. The first step is to locate the optima in the input space. For this purpose there are several search methods for computer experiments, [2,1,14,15,16]. After finding all the optima of interest, pattern search is run using the user inputs for the weights of the four measures and the grid points for each optimum to maximize each optimum's utility and determine its tolerance center. The optimum with the highest utility is the user's choice for the optimum that fits his/her criteria (weights).

Next we will illustrate an example in which the emulator is used to predict the test function surface for two minima, one with a large degree of skewness and one which is symmetric.

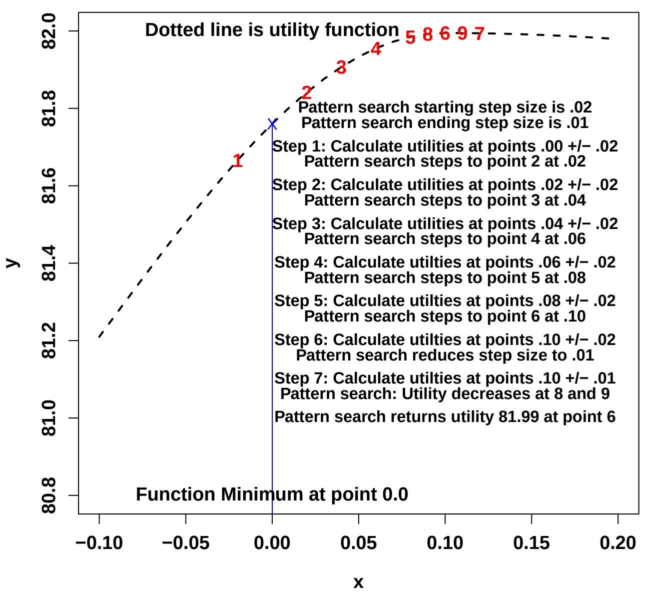

Figure 5. Schematic for Pattern Search: The dotted line is the utility function for a skewed test function in one dimension. The blue line with the blue 'X' at its intersection with the utility function indicates the minimum location and the utility at this location. Pattern search starts at the minimum point for the function and progresses to find the maximum of the utility function. The starting step size is set at .02 units. The ending step is set to .01 units. At each step it accesses the the utility function to compute the utilities at $x \pm .02$ where x is the current base location. Then it moves to whichever x value has the higher utility for its next base location. If the two points for which the utility is computed show no increase in utility, the step size is reduced by a factor of 2. When the ending step size is reached and both computed utilities show no increase in utility, pattern search ends and returns the maximum utility and its location. The graph shows the computed points in red numbers and the seven steps executed to find the maximum utility.

Table 1. Pattern Search Processing Steps: Pattern search starts with the starting point and uses the utility function to compute the utilities of four points: $(x_1 - size, x_2), (x_1 + size, x_2), (x_1, x_2 - size), (x_1, x_2 + size)$, where $size$ is used for step size. If any two of these points along different variable axes are greater than (x_1, x_2), it also computes the diagonal point corresponding to these two points. For example, if both utilities of $(x_1 + size, x_2)$ and $(x_1, x_2 + size)$ are greater than (x_1, x_2), then it computes the utility of $(x_1 + size, x_2 + size)$. In the table, the point number for the current coordinates is given in the first column, the (x_1, x_2) coordinates are in the second and third columns, the Utility is in the fourth column, the point number used as the base point for pattern search is in the fifth column, and the step size for the current step is in the sixth column. The step size changes only when the processing step does not produce a greater utility than the current base location. The last column indicates whether a diagonal point is computed. Altogether, there are 9 processing steps. A new step occurs when the base point changes or when the step size changes. Note that the utility is maximized at $(0.0475, 0.0475)$, the 38^{th} point, which has the utility 98.91%.

Point	x_1	x_2	Utility	Base Point	step	step size	diagonal point
1	0.0000	0.0000	81.09077	1	1	0.02	no
2	-0.0200	0.0000	76.79911	1	1	0.02	no
3	0.0200	0.0000	85.04911	1	1	0.02	no
4	0.0000	-0.0200	76.76875	1	1	0.02	no
5	0.0000	0.0200	85.09375	1	1	0.02	no
6	0.0200	0.0200	89.05208	1	1	0.02	yes
7	0.0000	0.0200	85.09375	6	2	0.02	no
8	0.0400	0.0200	92.67708	6	2	0.02	no
9	0.0200	0.0000	85.04911	6	2	0.02	no
10	0.0200	0.0400	92.73601	6	2	0.02	no
11	0.0400	0.0400	96.36101	6	2	0.02	yes
12	0.0200	0.0400	92.73601	11	3	0.02	no
13	0.0600	0.0400	97.09018	11	3	0.02	no
14	0.0400	0.0200	92.67708	11	3	0.02	no
15	0.0400	0.0600	97.98750	11	3	0.02	no
16	0.0600	0.0600	98.71667	11	3	0.02	yes
17	0.0400	0.0600	97.98750	16	4	0.02	no
18	0.0800	0.0600	98.46667	16	4	0.02	no
19	0.0600	0.0400	97.09018	16	4	0.02	no
20	0.0600	0.0800	98.69167	16	4	0.02	no
21	0.0500	0.0600	98.84167	16	5	0.01	no
22	0.0700	0.0600	98.59167	16	5	0.01	no
23	0.0600	0.0500	98.72917	16	5	0.01	no
24	0.0600	0.0700	98.70417	16	5	0.01	no
25	0.0500	0.0500	98.85417	16	5	0.01	yes
26	0.0400	0.0500	98.00000	25	6	0.01	no
27	0.0600	0.0500	98.72917	25	6	0.01	no
28	0.0500	0.0400	97.21518	25	6	0.01	no
29	0.0500	0.0600	98.84167	25	6	0.01	no
30	0.0450	0.0500	98.85417	25	7	0.005	no
31	0.0550	0.0500	98.79167	25	7	0.005	no
32	0.0500	0.0450	98.08631	25	7	0.005	no
33	0.0500	0.0550	98.84792	25	7	0.005	no
34	0.0475	0.0500	98.90625	25	8	0.0025	no
35	0.0525	0.0500	98.84375	25	8	0.0025	no
36	0.0500	0.0475	98.85938	25	8	0.0025	no
37	0.0500	0.0525	98.85312	25	8	0.0025	no
38	0.0475	0.0475	98.91146	25	8	0.0025	yes
39	0.0450	0.0475	98.85938	38	9	0.0025	no
40	0.0500	0.0475	98.85938	38	9	0.0025	no
41	0.0475	0.0450	98.13839	38	9	0.0025	no
42	0.0475	0.0500	98.90625	38	9	0.0025	no

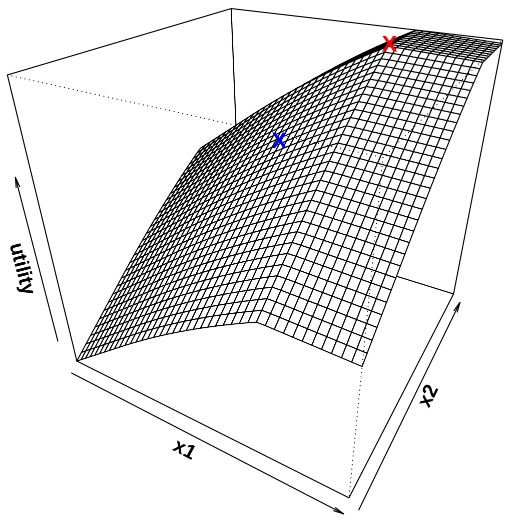

Figure 6. Perspective of Utility Function: The perspective of the utility function is shown here for the grid points in the interval $[-.1 \le x_1 \le .1, -.1 \le x_2 \le .1]$. It shows that the utility function is related in shape to the test function. However, its maximum is displaced to the right by about .05 and upward by about 0.05 units. This is one half the tolerance region side of 0.1. The starting point for pattern search is the blue 'X'. The ending point for pattern search is approximately located by the red 'X'.

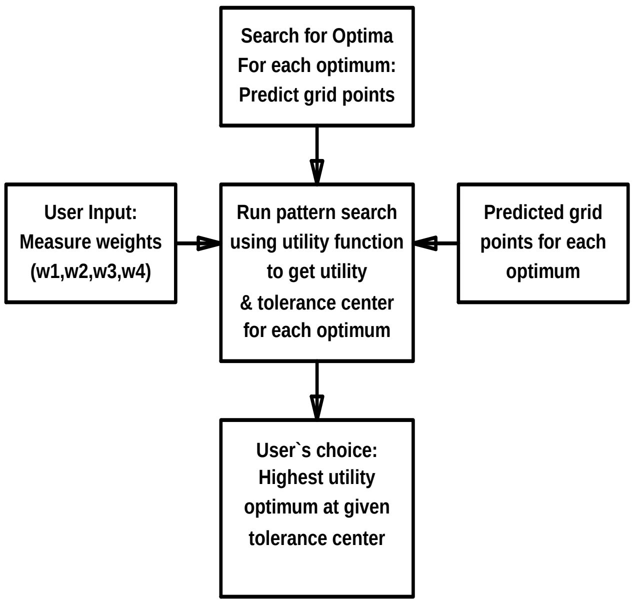

Figure 7. Algorithm Steps: As each optimum is found, the emulator predicts a grid of points centered at the optimum point with sides three times the tolerance region side. This data is stored for the next step. In the next step after the search is concluded, each optimum's utility is determined by running pattern search with the user supplied weights w and the stored grid points for the given optimum. The optimum with the highest utility is the user's choice. The tolerance center is the point achieving the maximum utility for the given optimum.

3 Illustration of the Algorithm with the Emulator

The example given in the last section shows how the algorithm works when the function is known. The example shown here compares two minima for a test function when the emulator is used to predict the test function surface. The perspective of this test function is shown in Figure 8. Mathematically, this function is:

$$f(x_1, x_2) = -[8 \cdot \text{Gamma}(x_1 - 0.35, \alpha = 1.02, \beta = 1) \cdot \text{Norm}(x_2, \mu = 1, \sigma = 0.2) + \\ \text{Norm}(x_1, \mu = .1, \sigma = 0.12) \cdot \text{Norm}(x_2, \mu = 1, \sigma = 0.1)] \tag{8}$$

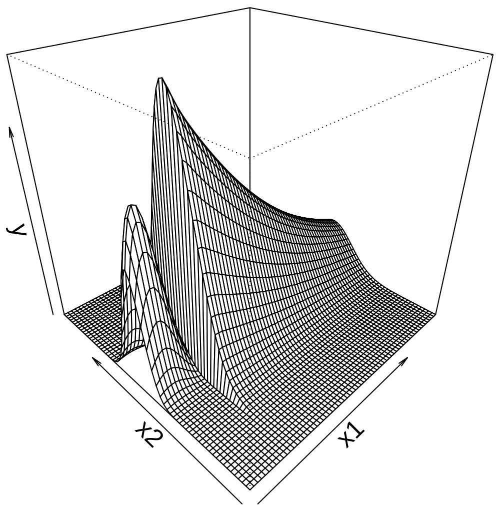

Figure 8. Two Minima Test Function: The perspective shows the negative of the function so the two minima appear as peaks for easier visualization. The domain of the function is $[0 \le x_1 \le 2], [0 \le x_2 \le 2]$. The symmetric minima occurs approximately at the point (.1,1.0). The larger skewed minimum occurs approximately at point (.357,1.0).

The first step in the processing is to run an optimization program with the emulator to find the locations and values for the two optima of the test function. For this purpose, the cluster search algorithm in [2] is used. After finding an optimum, to aid in the accuracy of the predicted points near an optimum, a few new training points (about $10 \cdot d$ where d is the input space dimension) are added in the region centered at the optimum point with sides three times the tolerance region side of .04 (in this case). These points are chosen from a Latin hypercube sample covering the region to maximize their distance from the existing training points. Then the emulator is invoked for each optimum to predict a grid of 15 by 15 surface points in this region. Each predicted point has 250 iterations of output values (this number depends on parameters given to the emulator) which are used to determine the optimum's utility. The difference between the utility function used in the case where the emulator provides predictive iterations and the case where the actual function values are known is that the there are 250 sets of predicted measures which are used to get a Monte Carlo estimate for the utility. Provided with the iterations for the grid of points, pattern search is invoked, starting with the optimum point and its utility, to step to the maximum utility point as described in Section 2. Table 2 shows the results.

What is noticeable in Table 2 is that the tolerance center point has been moved from the optimum point for both optima. The first optima's tolerance center has been moved more than the second optima's tolerance center. This is due to the asymmetry or skewness of the optimum. To show the advantage of this movement, the utility program is run subsequently with the utility values being determine by the starting or optimum points. These results are in Table 3. Notice that the symmetric optima (optimum

Table 2. Resulting Utility Data: The weights of the four measures are made equal: $w = (.25, .25, .25, .25)$. The seven pattern search steps for the the larger skewed optimum and the six pattern search steps for the smaller symmetric optimum give the results in the table. The optimum number is the first column. The optimum points (x_1, x_2) are in the second two columns. The fourth column has the optimum value, the fifth and sixth columns have the (x_1, x_2) center for the tolerance region having the best utility, the seventh column has the minimum value for the tolerance region, the eighth column has the average value for the tolerance region, the ninth column has the maximum value for the tolerance region, and the final column has the utility value.

Optimum	Opt x_1	Opt x_2	Opt Value	Tol x_1	Tol x_2	y_{min}	$y_{average}$	y_{max}	Utility
1	0.357	1.000	-15.851	0.369	1.005	-15.850	-15.629	-15.321	97.71%
2	1.000	1.000	-13.263	0.104	1.005	-13.263	-13.141	-12.940	84.77%

number 2) data changes only slightly. The utility is only about .1% smaller. However, there are significant changes for the skewed optima. One can see that there is a large effect due to the range of the output values. The utility is reduced by about 57 percentage points. This shows that the choice of the tolerance region center can be very important for skewed and asymetric optima.

Table 3. Utilities without Pattern Search Maximization: One utility computation is done for each optimum at the optimum point. The optimum number is the first column. The optimum points (x_1, x_2) are in the second two columns. The fourth column has the optimum value, the fifth and sixth columns have the (x_1, x_2) center for the tolerance region (same as the optimum point), the seventh column has the minimum value for the tolerance region, the eighth column has the average value for the tolerance region, the ninth column has the maximum value for the tolerance region, and the final column has the utility value.

Optimum	Opt x_1	Opt x_2	Opt Value	Tol x_1	Tol x_2	y_{min}	$y_{average}$	y_{max}	Utility
1	0.357	1.000	-15.851	0.357	1.000	-15.850	-10.084	-1.522	40.88%
2	1.000	1.000	-13.263	0.100	1.000	-13.263	-13.099	-12.931	84.66%

Another question that can be raised is how do the utilities compare to the utilities if the test function's actual values are used for the grid points. This depends on the accuracy of the emulator predictions. This accuracy is closely associated with the standard deviations of the predicted points [17]. The fit of the surface predicted points can also be ascertained from the errors in the points that are added to increase the accuracy of the emulator model at each optimization step. If these errors are small and getting smaller, a good fit of the emulator model to the actual function is probable. A run was made to use to test function values for the grid points. The resulting data is shown in Table 4. There are some minor differences from Table 2. The tolerance centers are the same, and, the utilities are the same. Therefore, the emulator model matches the actual test function very well.

Table 4. Utilities Using Test Function Grid Values: The utility computation is done using the actual function grid point values. The optimum number is the first column. The optimum points (x_1, x_2) are in the second two columns. The fourth column has the optimum value, the fifth and sixth columns have the (x_1, x_2) center for the tolerance region, the seventh column has the minimum value for the tolerance region, the eighth column has the average value for the tolerance region, the ninth column has the maximum value for the tolerance region, and the final column has the utility value.

Optimum	Opt x_1	Opt x_2	Opt Value	Tol x_1	Tol x_2	y_{min}	$y_{average}$	y_{max}	Utility
1	0.357	1.000	-15.851	0.369	1.005	-15.851	-15.625	-15.340	97.77%
2	1.000	1.000	-13.263	0.104	1.005	-13.263	-13.141	-12.940	84.77%

4 Conclusions

We propose an algorithm for efficiently determining the utilities of the multiple optima in the input space. The optimum with the highest utility is the best choice for the experiment. This algorithm makes use of Bayesian decision theory. It quantizes the attributes of interest in optimum selection, the optimum's smoothness and value, and takes into account the user's specific needs. It locates the tolerance region center which gives the best utility for both symmetric and asymmetric optima. The four measures used to quantize the attributes of interest are obtained from the predictions of a statistical emulator model in the tolerance region. Since this emulator model covers a small region relative to the input space it can be made accurate with a small number of training points. In other words, it makes efficient use of function evaluations. While it works in combination with a search algorithm for the optima, the selection methodology can also be employed on a preexisting set of function evaluations.

Other methods, although having value for determining an optimum's desirability, fall short by not automating the user's specific needs or by not considering some important measures that affect the optimum's usefulness for the purposes of the experiment. Ranking optima by their mean in some specified region around the optimum location does not take into account other important measures that concern the user [4]. Selecting optima based on a utility determined by their mean and variance does not take into account that, by moving the tolerance region from the optimum point, its utility may be greatly increased [3]. Selecting optima by their expected value in a confidence interval restricted by a variance limitation does not account for the accuracy of the influential variables [5]. And, again, the expected value is just the mean value which does not account for the minimum, maximum or range. Our method differs from other methods in that the user chooses what is most important for his/her optimization experiment by weighting four readily interpretable measures. The user's choice is then translated into a utility for each optimum making his/her choice apparent.

References

1. J. Guenther, H. Lee, and G. Gray, "Finding and choosing among multiple optima," *Journal of Applied Mathematics*, vol. **5**, 2014.
2. J. Guenther and H. Lee, "Cluster search algorithm for finding multiple optima," *Journal of Applied Mathematics*, vol. **7**, pp. 736–752, 2016.
3. V. Kroll, H. Levy, and H. Markovitz, "Mean-variance versus direct utility maximization," *The Journal of Finance*, vol. **39**, no. 1, pp. 47–61, 1984.
4. T. Sriver, J. Chrissis, and M. Abramson, "Pattern search ranking and selection algorithms for mixed variable simulation-based optimization," *European Journal of Operational Research*, vol. **198**, no. 3, pp. 878–890, 2009.
5. J. Kleijnen, *Design and Analysis of Simulation Experiments, Second Edition.* Springer, New York, 2015.
6. A. Saltelli, M. Ratto, T. Andres, F. Campolongo, J. Cariboni, D. Gatelli, M. Saisana, and S. Tarantola, *Global Sensitivity Analysis.* John Wiley and Sons Ltd., 2008.
7. J. Guenther, H. Lee, and G. Gray, "Sequential design for achieving estimated accuracy of global sensitivities," *Journal of Applied Stochastic Models in Business and Industry*, vol. **31**, no. 6, pp. 782–800, 2015.
8. R. Gramacy, M. Taddy, and S. Wild, "Variable selection and sensitivity analysis using dynamic trees, with an application to computer code performance tuning," *Annals of Applied Statistics*, vol. **7**, no. 1, pp. 51–80, 2013.
9. A. Gelman, J. B. Carlin, H. S. Stern, and D. B. Rubin, *Bayesian Data Analysis.* Chapman & Hall, 2004.
10. J. Berger, *Statistical Decision Theory and Bayesian Analysis.* Springer-Verlag, 1985.
11. T. Santner, B. Williams, and W. Notz, *The Design and Analysis of Computer Experiments.* Springer, 2003.
12. R. Gramacy and H. Lee, "Bayesian treed gaussian process models with application to computer modeling," *Journal of the American Statistical Association*, vol. **103**, no. 483, pp. 1119–1130, 2008.
13. G. Gray and T. Kolda, "Appspack 4.0, asynchronous parallel pattern search for derivative-free optimization," Sandia National Laboratories, #SAND 2009-0260P, Tech. Rep., 2008.
14. M. Taddy, H. Lee, G. Gray, and J. Griffin, "Bayesian guided pattern search for robust local optimization," *Technometrics*, vol. **51**, pp. 389–401, 2009.
15. H. Lee, R. Gramacy, C. Linkletter, and G. Gray, "Optimization subject to hidden constraints via statistical emulation," *Pacific Journal of Optimization*, vol. **7**, no. 3, pp. 467–478, 2011.
16. S. M. Wild and C. A. Shoemaker, "Global convergence of radial basis function trust-region algorithms for derivative-free optimization," *SIAM Review*, vol. 55, no. 2, pp. 349–371, 2013.

17. L. Bastos and A. O'Hagan, "Diagnostics for gaussian process emulators," *Technometrics*, vol. **51**, pp. 425–438, 2009.