

An Algorithm for Identification of Locally Optimal Basins in Large Dimensions on a Multi-Model Response Surface

Santosh Tiwari, Don Jones, and Simon Xu
General Motors Co.

ABSTRACT

Response Surface Models are often used as a surrogate for expensive black-box functions during optimization to reduce computational cost. Often, the CAE analysis models are highly nonlinear and multi-modal. A response surface approximation of such analysis as a result is highly multi-modal; i.e. it contains multiple local optima. A gradient-based optimizer working with such a response surface will often converge to the nearest local optimum. There does not exist any method to guarantee a global optima for non-convex multi-modal functions. For such problems, we propose an efficient algorithm to find as many distinct local optima as possible. The proposed method is specifically designed to work in large dimensions (about 100 ~ 1000 design variables and similar number of constraints) and can identify most of the locally optimal solutions in a reasonable amount of time. We also present a method to analyze natural partitions in a noisy data set and use it to characterize the locally optimal basins found by our method.

CITATION: Tiwari, S., Jones, D., and Xu, S., "An Algorithm for Identification of Locally Optimal Basins in Large Dimensions on a Multi-Model Response Surface," *SAE Int. J. Mater. Manf.* 8(2):2015, doi:10.4271/2015-01-0480.

INTRODUCTION AND MOTIVATION

A majority of CAE analysis models are highly non-linear and are generally computationally expensive. A single simulation (function evaluation) of such models may take anywhere from few hours to few days [1, 2, 3, 4, 5]. Furthermore, optimization using such models typically involves few hundred design variables and few hundred responses. The design variables typically correspond to parameters that can be tuned by the optimization algorithm. Example of such tunable parameters are gauges, material property, shape parameters, etc. These tunable parameters are inputs to the CAE simulation models. Responses are the outputs from the CAE simulation models that typically correspond to different types of load cases such as dynamics, crash, NVH, Stiffness, etc. Optimization with such analysis is generally very expensive and time consuming.

To improve the efficiency (reduce the total number of CAE simulations) of the optimization process, response surface models (RSM) have been widely used as surrogate for expensive black-box functions [6, 7, 8, 9, 10, 11]. Use of response surface models generally reduces the total number of function evaluations, thereby reducing the total time taken by the optimization process. Often the CAE analysis and the response surface models are highly non-linear and exhibit non-convexity and multimodality. A gradient-based optimizer will in such cases converge to the nearest local optimum.

It is desired to improve the search capability of the optimizer by improving the probability of discovering the global optimum (by identifying as many local optima as possible). Additionally, with increase in computational resources, it is possible to evaluate multiple designs in parallel. In such a case, it is desired to exploit the available

computing power to improve the efficiency of the optimization process by evaluating multiple designs in parallel. Motivated by the opportunity to exploit parallelism and a desire to improve the global search capability, the following research questions are posed and answered in this paper.

1. Is the response surface multi-modal? This question can be alternatively posed as: Is the solution to the optimization problem sensitive to the starting point? If the obtained solution depends upon the starting point, then the response surface is likely multi-modal, and we may only be performing a local search. Thus, the first research question is: how to identify if the response surface is multi-modal?
2. Suppose it is determined that the response surface is multimodal. In such a case, can we find all (or as many as possible) locally optimum solutions in a reasonable amount of time?

OPTIMIZATION PROCEDURE

Due to the computationally expensive nature of the CAE simulation models, a response surface model is used as a surrogate on which the proposed algorithm is applied. It is assumed that no prior information is available about the CAE simulation models and they are treated as black-box simulation codes. The following is the schematic of the entire optimization process.

1. Generate an initial sample using a suitable Design of Experiments (DOE) technique. It is desired that the initial sample covers the entire search space. The sample size is a

user-definable parameter; however a suitable (default) value can be assigned depending upon i) the number of variables, ii) the computational expense of running the CAE simulations, and iii) available computing resources. The design variables are denoted by the vector X .

2. Perform CAE simulations in parallel or in batches to generate the response values corresponding to each design point. The responses are denoted by the vector Y .
3. Fit a response surface model (RSM) such as Kriging, Radial Basis Function, etc. to the generated data. The response surface creates a mapping from inputs X to outputs Y . It is desirable that the RSM interpolates the data points (i.e. the prediction error is zero at the actual data points). The prediction error generally increases as we move away from the data points. It should be noted that interpolating the data points may not always be the best approach. If the sample is dense, then a regression type fit which follows the general trend may be more suitable for optimization. It is also desirable that the response surface provide an estimate of gradients along with the response values.
4. Run the proposed algorithm (discussed in the next section) on the generated response surface to identify as many distinct local optima as possible.
5. Typically a large number of local optima are found. Select a subset of the local optima that are found and run CAE simulations at those points to get the correct response values.
6. Include these new points in the sample of DOE points and refit the response surface and repeat the steps (3, 4, and 5) until some termination criteria is met.

Step 4 in the above procedure constitutes the proposed algorithm which is discussed in the next section. It should be noted that the points identified in step 4 are locally optimal on the response surface. If the response surface is accurate, then these points will also be locally optimal for actual CAE simulations. Alternatively, if the response surface has error, then these are the points at which the response surface needs to be fixed since they will act as traps for any local optimizer. Thus, it can also be stated that the proposed optimization procedure fits a response surface to CAE simulations such that the surface is exact (has very small prediction error) at the locally optimal points. Such a response surface generated at the end of optimization can be used to analyze the properties of the CAE responses if desired. In step 5, only a subset of locally optimal solutions are used to run the CAE analysis because large locally optimal basins attract a large number of points and only one point from the pool can be used for CAE analysis. Furthermore, after few iterations, the approximation error at some of the locally optimal basins is small enough that no further refinement is necessary in those regions. As iterations proceed, and surface gets refined, new locally optimal basins are discovered and the response surface is made more accurate in those areas. The optimization process can be terminated if one of the following criteria is met.

1. A given number of function evaluations (the computational budget) has been exhausted.

2. No new locally optimal basins are discovered and response surface has very small error at the locally optimal design points.
3. If there is a target for design objective, then the process can be terminated if that target is met.

PROPOSED ALGORITHM

The proposed algorithm requires as input the problem information (number of constraints, their bounds, etc.), and the allowed computational budget which can be specified either in terms of number of function evaluations on the response surface or the total computational time. It should be noted that even if a response surface is very cheap to evaluate, it may take considerable time to perform multiple optimization runs in large dimensions and therefore the allowed computational budget affects the number of locally optimal basins discovered by the proposed algorithm.

Various approaches have been proposed in literature for global optimization and identify locally optimal solutions. Niching based genetic algorithms (GAs) [12, 13, 14] have been successfully applied to solve such problems. By using a sharing function which degrades the objective function value based on crowding, niching based GAs can maintain stable populations of diverse individuals. An alternative to sharing function approach is the fuzzy evaluation of objectives by means of ϵ -domination to maintain stable populations [15, 16]. There are several other approaches [17, 18, 19, 20] to finding global optimum to multi-modal functions. However, all these approaches are non-deterministic in nature and rely on chance to discover the globally optimal solutions. The obtained solutions depend upon the random seed and in general, multiple runs with different random seeds need to be performed to improve the probability of discovering the global optimum. The algorithm proposed in this paper is deterministic (does not depend upon a random number to initiate the optimization process) and involves systematic step-by-step process to identify locally optimal solutions.

The proposed algorithm begins by generating a uniformly distributed sample of points using strength two orthogonal arrays. The number of levels can be different for each variable and is determined by the user. Typically an orthogonal array generates a prohibitively large number of samples in large dimensions. To address this issue, *k-means* clustering is used to prune the obtained sample to a manageable size. If the initial sample is uniformly distributed and dense enough, then it is assumed that there would be at least one point in most of the locally optimal basins. Increasing the sample size therefore increases the probability of discovering very narrow or small locally optimal basins.

A local gradient-based optimization is then performed for each obtained sample. After this step, a large number of local optima are obtained. If a locally optimal basin is very large then there will be several locally optimal solutions trapped inside that basin. We then use cophenetic distance measure and Silhouette value analysis to identify natural partitions in the obtained data set. Based on this

analysis, the correct number of clusters are identified and the results are presented using a dendrogram. These clusters represent the locally optimal basins. The following is the step-by-step description of the proposed optimization algorithm. Step 1: Generate a uniformly distributed random sample (Figure 1). The suitable sample size is either specified by the user or is automatically determined based on the dimensions of the search space.

- Step 2: Perform a local gradient-based optimization for each point in the sample to obtain locally optimal solutions (Figure 2).
- Step 3: Perform Hierarchic Agglomerative Clustering (HAC) and Silhouette value analysis to group the point into clusters and also identify the correct number of clusters based on natural partitions in the data (Figure 3). Present the obtained solutions using a dendrogram tree for easy visualization.
- Step 4: Characterize each cluster based on its size and the number of points in that cluster. Identify a representative solution from each cluster (Figure 4).
- Step 5: (Optional) Prune the number of obtained solutions to a desired size (Figure 5) that can be used for CAE simulation runs.

This proposed algorithm is fully automated and no human intervention is necessary. We now discuss each step of the proposed algorithm in detail.

Test Problem used for Demonstration

The test problem used for this study has 160 design variables with box constraints and 114 non-linear inequality constraints. It takes roughly 30 ms to perform a single evaluation on the response surface; whereas it takes about 24 hrs for a single CAE simulation on a compute cluster with multiple nodes. It should be noted that the dimension of the search space is 160 which is very large for any meaningful sampling. Even if we were to generate 160,000 samples, it would still be equivalent to $\sqrt[160]{160,000} \approx 1.08$ points per dimension. A random sample with 160,000 design points was generated. It was found that none of the 160,000 solutions were feasible (i.e. all the 114 non-linear inequality constraints were never simultaneously satisfied). An evolutionary algorithm working with such a problem will have difficulty finding a feasible solution owing to its random search strategy.

Generation of the Initial Sample (Step 1)

It is desired that the initial sample has the following properties.

- It is free from any bias (unbiased sampling).
- It is as uniform as possible.
- It spans all the dimensions equally.
- It is spread throughout the search space.

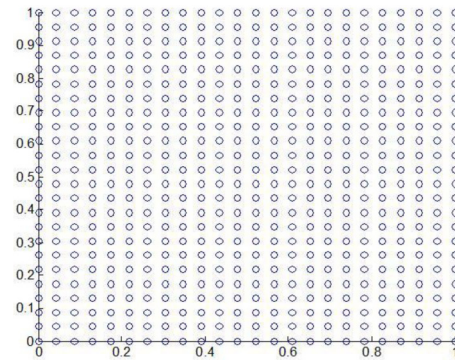


Figure 1. Generate Initial Sample (step 1)

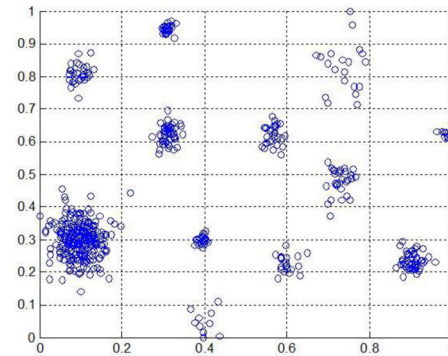


Figure 2. Perform Local Optimization (step 2)

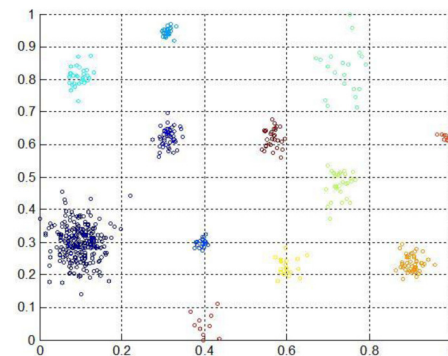


Figure 3. Perform HAC (step 3)

In 160 dimensions, it is practically impossible to uniformly sample the entire search space. Even with just two points per dimension, the total number of points would be $2^{160} \approx 1.46 \times 10^{48}$. Almost all the systematic sampling methods (e.g. orthogonal arrays) generate a very large number of points for 160 dimensions. Owing to this limitation, we resort to alternative methods to generate the initial sample. These alternative methods are either random or heuristic in nature, and therefore their performance needs to be compared to identify the most suitable technique. Owing to 160 dimensions, the chosen sample size is 1600 (ten times the number of variables) for this study. Later in this paper, we investigate a suitable value for the sample size. For now, we start our investigation by setting the sample size to 1600. The following sampling methods are investigated.