

The Use of Surrogate Modeling Algorithms to Exploit Disparities in Function Computation Time within Simulation-Based Optimization

Michael J. Sasena, Panos Y. Papalambros and Pierre Goovaerts

University of Michigan, Ann Arbor, MI 48109-2125, USA
{msasena, pyp, goovaert}@engin.umich.edu

Abstract

Expensive computer simulations pose interesting challenges in design optimization. Conservation of function evaluations and the ability to deal with noisy responses become prime concerns. A common approach is to create an inexpensive approximation that may also smooth out some of the small scale noise. Typically, an approximation is made for the objective function and each of the constraint functions, which is inefficient if some of the functions (either objective or constraint) are *not* expensive. This paper presents modifications to the surrogate model-based optimization algorithm, EGO. By incorporating information from inexpensive functions, analytical tests required between 10% and 50% fewer iterations and 20% to 65% less time. A simulation-based case study showed greater benefits.

Keywords: Approximations, Computer Simulations, Optimization

1. Introduction

In simulation-based optimization, the expense of the analysis models is often a prime concern. While surrogate modeling algorithms can reduce computational time, not all such algorithms can exploit differences in the computational costs of functions in the problem statement. In previous work by Nelson and Papalambros [1], inexpensive functions were used directly in a trust region algorithm, resulting in fewer expensive analyses. This provides motivation to extend the work to surrogate model-based optimization. The algorithm examined in this work, Efficient Global Optimization (EGO) [2], utilizes kriging models [3] as global approximations to the expensive functions. The algorithm works as follows:

1. Use space-filling Design of Experiments to obtain initial sample of true functions.
2. Fit kriging model to each function.
3. Use models to determine candidate location(s) most beneficial to sample next according to an infill sampling criterion.
4. Sample point(s) of interest on true functions and update kriging models.
5. Check for termination, otherwise return to 2.

Originally, EGO was designed for cases where all the functions were expensive. The approach here does not have such limitations. It is not always clear whether to classify a simulation response as expensive or inexpensive, nor is it always feasible to segregate the two classes of functions into independent simulation runs. However, if one can do this, then eight types of well-defined optimization problems exist for the various combinations of expensive and inexpensive objective and constraint functions, see Figure 1.

The changes proposed here enable EGO to differentiate between all eight problem types, including the special case where all functions are considered inexpensive (types 3 and 4). Because DIRECT, another global optimization algorithm [4], is used to solve the internal optimization problem of EGO step 3, EGO degenerates into DIRECT for those problem types. This paper focuses on the more interesting cases of types 5 through 8, where both inexpensive and expensive functions exist. We propose using information from the inexpensive functions to avoid sampling the expensive functions when it is not beneficial to do so. The methods for making decisions are referred to as filters. Table 1 lists the eight problem types and the method that shall be used to solve each. Expanding EGO to handle the entire array of possibilities allows the designer to solve the optimization problem efficiently, regardless of the cost of the functions. Because EGO is now able to handle a *superset* of the types of problems it was originally designed for, the algorithm using the modifications described below will be referred to as *superego*.

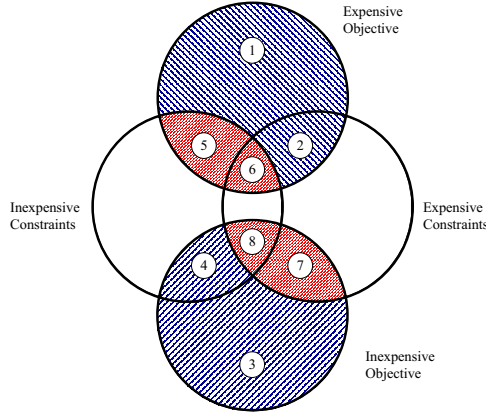


Figure 1: Possible Problem Statement Types

Table 1: Problem Types and Associated Solution Strategies

Type	Objective Function	Constraint Functions	Solution Strategy
1	Expensive	None	Unconstrained EGO
2	Expensive	Expensive	Constrained EGO
3	Inexpensive	None	Unconstrained DIRECT
4	Inexpensive	Inexpensive	Constrained DIRECT
5	Expensive	Inexpensive	Filter Type I
6	Expensive	Both expensive and inexpensive	Filter Type II
7	Inexpensive	Expensive	Filter Type III
8	Inexpensive	Both expensive and inexpensive	Filter Type IV

2. Filter Method

One must first distinguish between the original optimization problem to be solved and the inner loop optimization problem that forms Step 3 of the EGO algorithm. For clarity, we shall refer to the optimization problem of Step 3 as the Infill Sampling Criteria (ISC) problem. EGO was primarily developed for unconstrained optimization. To handle constraints, Schonlau proposed multiplying the ISC objective by the probability of feasibility for each constraint function [2]. This method had its weaknesses, as did the attempt to use a penalty method on the ISC [5]. In this paper, we solve the ISC problem as a constrained optimization problem, eliminating many of the difficulties in locating constraint boundaries.

Our work takes advantage of the fact that a constrained ISC problem can be solved with a global searching, derivate-free algorithm, namely DIRECT. The ISC problem of EGO is solved as

$$\begin{aligned} \min f &= ISC(\mathbf{x}) \\ \text{subject to: } &\hat{g}_i(\mathbf{x}) \leq 0, \forall i \end{aligned} \quad (1)$$

where $\hat{g}_i(\mathbf{x})$ refers to the kriging model prediction of constraint g_i at \mathbf{x} .

Information from any inexpensive functions is used here to prevent suboptimal points from being sampled on the expensive true functions in Step 4 of EGO. This is done by imposing two additional constraints on the ISC problem of equation 1. The first checks if any inexpensive constraint functions are infeasible, and the second checks if an inexpensive objective function is worse than the current best design point. The original constraint of keeping the predicted value of the expensive constraints feasible is retained as well. Thus the Filter Method changes the ISC problem to the following:

$$\begin{aligned}
& \min f = ISC(\mathbf{x}) & (2) \\
& \text{subject to: } \hat{g}_i(\mathbf{x}) \leq 0, i = 1, \dots, n_{expc} \\
& \quad g_j(\mathbf{x}) \leq 0, j = 1, \dots, n_{inexpc} \\
& \quad f_{inexp}(\mathbf{x}) \leq f_{min}^n
\end{aligned}$$

where n_{expc} and n_{inexpc} are the numbers of expensive and inexpensive constraints, respectively, f_{inexp} is the inexpensive objective function of the original optimization problem, and f_{min}^n is the best feasible point found thus far. There are four Filter Types based on which kinds of functions exist as shown in Table 1.

3. Analytical Examples

The effectiveness of the Filter Method was evaluated for a suite of two-dimensional analytical examples described in Table 2. Notice that examples 3 and 4 are the Three Hump Camel-back [6] and Goldstein Price [7] test functions, respectively, modified by adding constraints. The log of $f(\mathbf{x})$ was used in example 4 to improve the quality of the kriging models.

Table 2: List of Analytical Examples

Example no.	Variable Bounds	Solution
1	$x_i \in [0, 5] \forall i$	$f(2.75, 2.35) = -1.17$
2	$x_i \in [0, 5] \forall i$	$f(3.07, 2.10) = -0.55$
3	$x_1 \in [-3, 3], x_2 \in [-1.5, 1.5]$	$f(1.75, 0.87) = 0.30$
4	$x_i \in [-2, 2] \forall i$	$f(0.60, -0.40) = 5.67$
5	$x_i \in [0, 1] \forall i$	$f(0.20, 0.83) = -1.33$

Example 1

$$\begin{aligned}
\min f(\mathbf{x}) &= 2 + 0.01(x_2 - x_1^2)^2 + (1 - x_1)^2 + 2(2 - x_2)^2 + 7 \sin 0.5x_1 \sin 0.7x_1x_2 \\
\text{subject to: } g(\mathbf{x}) &: -\sin(x_1 - x_2 - \frac{\pi}{8}) \leq 0
\end{aligned}$$

Example 2

$$\begin{aligned}
\min f(\mathbf{x}) &= -\sin(x_1 - x_2 - \frac{\pi}{8}) \\
\text{subject to: } g(\mathbf{x}) &: -1 + 0.01(x_2 - x_1^2)^2 + (1 - x_1)^2 + 2(2 - x_2)^2 + 7 \sin 0.5x_1 \sin 0.7x_1x_2 \leq 0
\end{aligned}$$

Example 3

$$\begin{aligned}
\min f(\mathbf{x}) &= 2x_1^2 - 1.05x_1^4 + \frac{1}{6}x_1^6 - x_1x_2 + x_2^2 \\
\text{subject to: } g(\mathbf{x}) &: (3 - x_1)^2 + (1 - x_2)^2 - 3 \leq 0
\end{aligned}$$

Example 4

$$\begin{aligned}
\min f(\mathbf{x}) &= (1 + A(x_1 + x_2 + 1)^2)(30 + B(2x_1 - 3x_2)^2), \\
& \text{where } A = 19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2, \\
& \text{and } B = 18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2 \\
& \text{subject to: } g_1(\mathbf{x}) : -3x_1 + (-3x_2)^3 \leq 0 \\
& \quad g_2(\mathbf{x}) : x_1 - x_2 - 1 \leq 0
\end{aligned}$$

Example 5

$$\begin{aligned} \min f(\mathbf{x}) &= -(x_1 - 1)^2 - (x_2 - 0.5)^2 \\ \text{subject to: } g_1(\mathbf{x}) &: ((x_1 - 3)^2 + (x_2 + 2)^2) \exp(-x_2^7) - 12 \leq 0 \\ g_2(\mathbf{x}) &: 10x_1 + x_2 - 7 \leq 0 \\ g_3(\mathbf{x}) &: (x_1 - 0.5)^2 + (x_2 - 0.5)^2 - 0.2 \leq 0 \end{aligned}$$

To reduce the initial sample locations' influence on the results, EGO was run using a series of 50 initial 21-point (11-point in the case of example 5), random Latin Hypercube design of experiments (DOE). Using the same initial DOE's, superego was run for the various filter types by making different choices as to which functions were considered inexpensive. In all cases, the algorithm was stopped once it located a feasible point within 5% of the global minimum value. EGO and each filter type were then compared with respect to clock time and the number of iterations required (one function evaluation per iteration for this implementation). The kriging model parameters were held fixed for all iterations so that the choice of model fitting frequency did not impact the study. Only examples 4 and 5 ran Filter Types II and IV because multiple constraints are required. Two tests for each of those Filter Types were run by swapping the constraints considered to be expensive. For the following three tables, the results of superego are categorized as better, the same, or worse according to how many function evaluations were required compared to EGO. Table 3 summarizes the breakdown of these categories.

Table 3: Summary of Comparisons to EGO (in Percentage) for Each Category

Fn Call Comparison	Filter Type I	Filter Type II	Filter Type III	Filter Type IV
better	59.2	52.0	59.2	74.5
same	30.0	40.5	36.8	24.0
worse	10.8	7.5	4.0	1.5

There are several trends worth noting. For one, superego is *not* systematically better than regular EGO, yet is at least as good in the vast majority of cases. Also, Filter Types III and IV are worse than regular EGO less often than Filter Types I and II. This is most likely due to the fact that the former consider the objective function inexpensive. Using this information forces superego to find feasible points at least as good as the current best at each iteration, thereby reaching good solutions more quickly than for Filter Types I or II.

Tables 4 and 5 show the relative improvement (or worsening) in the number of iterations and time required to find the optimum. For each Filter Type, the average difference is shown alongside the average standard deviation across examples. The overall averages for each filter are weighted by the percentages of occurrence in each category as shown in Table 3. From Tables 4 and 5 one can observe that, on average, superego outperforms EGO for any given Filter Type, both in terms of function evaluations and time required. Also, Filter Types III and IV have both higher relative improvement and lower standard deviations for both time and iterations required. This is another indication that exploiting the information from an inexpensive objective function can consistently lead to significantly improved efficiency.

Table 4: Relative Improvement (in Percentage) over EGO in Iterations Required

Fn Call Comparison	Filter Type I		Filter Type II		Filter Type III		Filter Type IV	
	avg	std dev	avg	std dev	avg	std dev	avg	std dev
better	50.7	14.7	40.1	16.6	60.7	18.8	63.6	15.9
worse	-62.0	62.6	-151.2	229.0	-68.3	27.1	-44.4	7.9
overall	23.3	15.5	9.5	25.8	33.2	12.2	46.7	12.0

Table 5: Relative Improvement (in Percentage) over EGO in Time Required

Fn Call Comparison	Filter Type I		Filter Type II		Filter Type III		Filter Type IV	
	avg	std dev	avg	std dev	avg	std dev	avg	std dev
better	63.7	11.6	49.2	14.6	72.9	14.8	77.0	10.9
same	18.6	17.7	15.1	5.6	19.2	3.7	33.2	4.3
worse	-23.3	49.4	-126.0	231.8	-62.3	67.1	8.1	11.7
overall	40.7	17.5	22.3	27.3	47.7	12.8	65.4	9.3

4. Case Study

In this section, a simulation-based case study is presented using a vehicle system analysis package known as ADVISOR [8]. A product platform study [9] is performed with a premium compact (PC) and lower midsize (LM) vehicle. The two vehicles are to share a common engine while having independent final drive ratios. The premium compact is designed to maximize fuel economy while meeting six performance constraints. The lower midsize is a slightly larger vehicle designed to minimize 0 - 60 mph acceleration time while meeting 25 mpg fuel economy and performance constraints 10% more demanding than for the premium compact. This results in a three variable design problem with a bicriteria objective ($w_{PC} f_{PC} + w_{LM} f_{LM}$) function and 12 constraints.

On a 550 MHz Pentium III computer, each vehicle analysis required approximately 30 seconds for fuel economy and 7 seconds for performance calculations. Because the performance constraints were too expensive relative to the fuel economy, they were replaced with highly accurate spline models. Using the spline models as the “true” functions, five optimization studies were performed by varying the weights w_{PC} and w_{LM} between 0 and 1 such that $w_{PC} + w_{LM} = 1$. In addition, two optimization runs were performed to locate the so-called *null platform* where each vehicle is optimized independently. Filter Type II was used for superego on all cases except for $w_{PC} = 0$ where Filter Type IV was used.

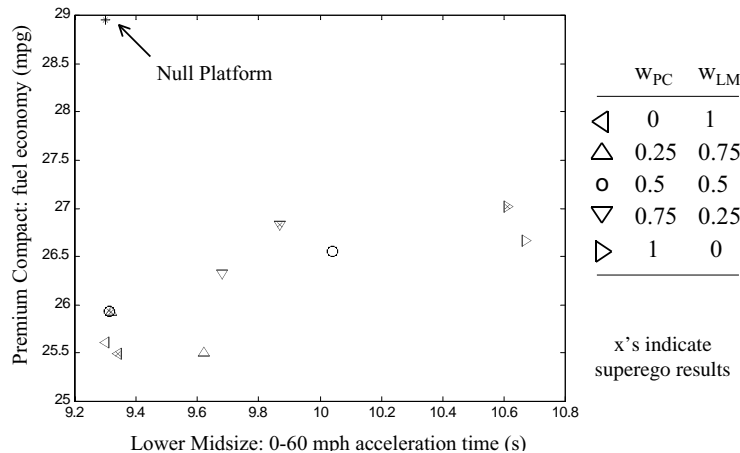


Figure 2: Null Platform and Pareto Set for Vehicle Case Study

The objectives for each vehicle are shown in the Pareto plot of Figure 2. Notice the large difference between EGO and superego for a given (w_{PC}, w_{LM}) combination. Superego found a somewhat better combined objective than EGO in all cases except the \triangleright test which was only 0.2% worse. Table 6 examines the time and number of iterations it took superego to locate a feasible point better than EGO’s best solution for a given optimization problem. In all but one case, the Filter Method significantly improves the efficiency of the algorithm.

5. Conclusions

This research attempted to quantify the gains in efficiency to a surrogate model optimization algorithm made by exploiting information from the inexpensive functions. By adding constraints to the ISC subproblem of Equation (1), the algorithm avoids evaluating locations known to be suboptimal. While this does not guarantee that the solution will be found more quickly, it tended to perform more efficiently most of the time.

Table 6: Improvements over EGO (in Percentage) for Vehicle Study

	Pareto 1	Pareto 2	Pareto 3	Pareto 4	Pareto 5	Null 1	Null 2
Iterations	-50.0	88.9	75.0	92.9	72.7	50.0	81.3
Time	-95.2	86.3	70.0	91.2	69.8	34.6	80.4

For the analytical tests, direct use of inexpensive functions decreased the number of required iterations by between 10% and 50% and the required time between 20% and 65%. In a simulation-based case study, the differences were more apparent. Neglecting the case where EGO required two fewer iterations, superego tended to reduce the number of iterations and time by 77% and 72%, respectively.

The results of this study point to new avenues of increased efficiency. The filter constraints used here are not the only kind one can use. For example, one may want to use information from the expensive functions to determine if a candidate infill sample location is in an area of poor model reliability. For some iterations, the desire to improve the expensive model functions locally could override the desire to sample only feasible and/or better locations. The methodology used here is generic enough to allow for such a constraint type. Another promising avenue is the ability of these filters to serve as convergence criteria, a current weakness of EGO. For example, one could constrain the ISC problem to improve upon the best known point according to some statistical confidence interval such as that proposed by Cox and John [10]. Failure to find a feasible solution to the ISC problem would thus terminate the algorithm.

One unanswered question arising from this research is how to best incorporate expensive constraints. Currently, the filter specifies $\hat{g}(\mathbf{x}) \leq 0$ for all expensive constraints. If the kriging model incorrectly predicts infeasibility, the filter then becomes overly restrictive, and it may slow down the algorithm. Another approach is to put the expensive constraints on a lower priority level in the filter via penalties or goal programming. Namely, first make sure that the inexpensive functions are not suboptimal then, if possible, ensure the predicted values for the expensive constraints are feasible. Some such adaptation may further improve the efficiency of superego.

6. References

- [1] Nelson S A II and Papalambros P Y. The use of trust region algorithms to exploit discrepancies in function computation time within optimization models. *ASME J of Mechanical Design*, 1999, 121(4): 552-556.
- [2] Schonlau M. *Computer Experiments and Global Optimization*. Doctoral thesis, University of Waterloo, Department of Statistics, Ontario, Canada, 1997.
- [3] Goovaerts P. *Geostatistics for Natural Resources Evaluation*. New York: Oxford University Press, 1997.
- [4] Jones D R, Perttunen C D and Stuckman B E. Lipschitzian Optimization Without the Lipschitz Constant. *J of Optimization Theory and Application*, 1993, 79: 157-81.
- [5] Sasena M J, Papalambros P Y and Goovaerts P. Exploration of Metamodeling Sampling Criteria for Constrained Global Optimization. *Engineering Optimization*. Paper accepted May 2, 2001.
- [6] Hardy J W. An implemented extension of Branin's method. In LCW Dixon and GP Szego (Eds.), *Towards Global Optimization*. pp 117-142. North-Holland, Amsterdam, 1975.
- [7] Pronzato L, Walter E, Venot A and Lebruchec J F. A general purpose global optimizer: Implementation and applications. *Mathematics and Computers in Simulation*, 1984, 26: 412-22.
- [8] National Renewable Energy Laboratory: ADVISOR homepage. <http://www.ctts.nrel.gov/analysis/>
- [9] Fellini R, Papalambros P and Weber T. Application of a Product Platform Design Process to Automotive Powertrains. *Proc 8th AIAA/NASA/USAF/ISSMO Symp on Multidisciplinary Analysis and Optimization*, AIAA-2000-4849, Long Beach, CA, USA, Sept 6-8, 2000.
- [10] Cox D D and John S. SDO: A Statistical Method for Global Optimization. In MN Alexandrov and MY Hussaini (Eds.), *Multidisciplinary Design Optimization: State of the Art*. pp 315-29. SIAM, Philadelphia, PA, 1997.