DEVELOPMENT OF A "MONTY HALL" ANALOG FOR HEURISTIC ALL-AT-ONCE OPTIMIZATION

K.F. Hulme
State University of New York at Buffalo
New York State Center for Engineering Design and Industrial Innovation (NYSCEDII)
5 Norton Hall
Buffalo, New York 14260-1810, USA

Keywords: Multidisciplinary Design Optimization, CASCADE, Monty Hall, heuristic optimization, simulated annealing, All-at-Once, SAND

Abstract

Past studies in Multidisciplinary Design Optimization (MDO) have shown that All-at Once (AAO) optimization can be an extremely intuitive and useful alternative means to approach the solution of a multidisciplinary analysis and optimization simultaneously. However, its utility has shown to decrease for larger problems with higher degrees of non-linearity and non-convexity. The present research presents a new heuristic optimization algorithm intended for solving coupled (multidisciplinary) design problems posed in the form of an AAO optimization. The hope is that the algorithm presented and developed herein can be used to improve upon past findings where conventional gradient-based optimization methods have been found to fail. The algorithm is modeled after the structure and decision process behind the famous "Monty Hall" problem, which gained its name from the host of the 1970's TV show, "Let's Make a Deal". The algorithm developed in this research has also been modeled after numerous other popular heuristic optimization algorithms which promote the concepts of exploration as well as exploitation of the design space, namely simulated annealing, tabu search, and genetic algorithms. The algorithm will first be presented on a small, simple, well-known test problem to most easily demonstrate its characteristics and assess its functionality. Thereafter, the algorithm will be implemented on two additional multidisciplinary system simulations of greater size and complexity, both of which will be generated using the previously developed CASCADE MDO simulation tool. For all test systems, the performance of the new method will be compared to other optimization approaches, such as gradient-based methods (using MS Excel's internal solver), simulated annealing, and a pure random search.

* Research Associate, Member AIAA

Introduction

Most large-scale engineering product and process design is multidisciplinary in nature. Essentially, this means that numerous inter-related design groups and/or design tasks are involved in the process of the overall final design. This process typically has numerous computationally expensive stages (such as sensitivity analysis, engineering analysis, and numerical optimization, to name a few) during each design cycle, each of which have their own internal stages of costly iteration. In the early 1980's, a field of research emerged which inherently attempts to unite the concepts of concurrent engineering with large-scale, multidisciplinary engineering design. This emerging field has since been coined Multidisciplinary Design Optimization, or MDO[17].

The general MDO approach is intuitive: divide a single large task into a grouping of smaller, interrelated (coupled), and more manageable sub-tasks. The large task is often referred to as a *system*, and the smaller, interrelated tasks are often referred to as *subsystems*. Each subsystem typically contains *design variables*, generically denoted as "X", which are parameters that might change during a formal optimization procedure. In addition, each subsystem typically also contains additional unknown outputs, often referred to as *behavior variables*, generically denoted as "Y". It is these variables that might change during a complex system analysis. Further, these variables represent the coupling links between the subsystems.

Background

Few multidisciplinary system decompositions result in a structure which is truly hierarchical (top-down) in nature. This inherent lack of hierarchy requires that the system analysis associated with the overall design cycle be initialized to some set of values, and iteratively converged thereafter. Subsequent to attaining a converged analysis solution, a sensitivity analysis is performed. The sensitivity analysis can be a numerical procedure such as finite differencing or the Global

Sensitivity Equation (GSE) method [4,18]. The sensitivity analysis is required for the optimization of the overall design. The optimization step itself will typically cause certain optimization variables to change, which then necessitates the re-convergence of the system analysis. Hence, the entire design cycle repeats itself until a converged solution is attained. A summary of such non-hierarchic design synthesis, conventionally referred to as "Multiple Discipline Feasible" (MDF) is illustrated in Figure 1.

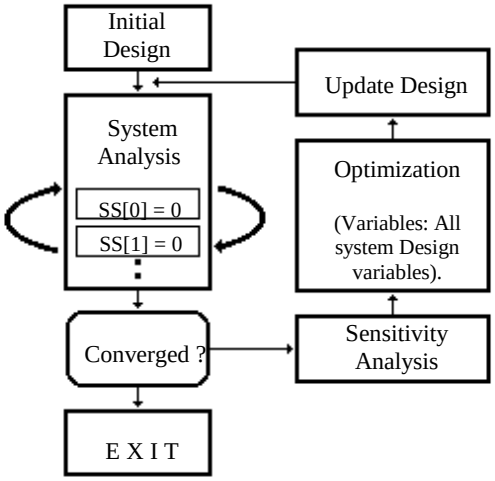

Figure 1: Non-hierarchic design synthesis - The "Multiple-Discipline-Feasible" (MDF) strategy

More recently, researchers have focused on alternate methods for posing and solving the multidisciplinary design problem. An approach has been developed which treats the entire multidisciplinary design cycle seen in Figure 1 as a single large optimization problem. This is accomplished by converting the system analysis equations into equality constraints, and by treating both system design variables and subsystem outputs (behavior variables) as optimization variables. Such an approach has been referred to in literature as both "Simultaneous Analysis and Design" (SAND) and "All-at-Once" (AAO) [5,6,1]. The primary advantage of AAO is the elimination of an iterative design cycle for attaining an optimal design through the outright elimination of costly iterative analysis evaluations. One possible disadvantage of AAO is that a much more complicated optimization problem results. More optimization variables and more equality constraints are present in the AAO formulation. These variables and equations stem from the addition of the system analysis equations to the optimization problem statement. A second disadvantage is that disciplinary feasibility is only attained at a relative or at an absolute extremum. This reduces the possibility of attaining a valid design solution if the optimizer is unsuccessful in attaining the global optimum solution. A generalized summary of the AAO strategy is seen in Figure 2. Notice that the "Residual Evaluator" has replaced the iterative System Analysis seen in Figure 1. In the Residual Evaluator, the analysis equality constraints are posed.

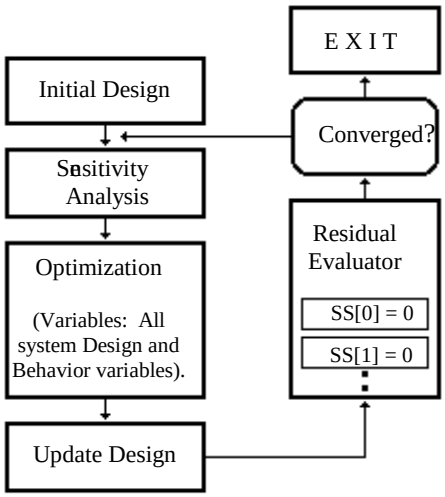

Figure 2: The "All-at-Once" (AAO) strategy

Clearly, the optimization component is a major aspect in any solution strategy that is implemented, especially the AAO approach, whose entire cycle is essentially one all-encompassing optimization step. Though gradient-based techniques are extremely useful in certain situations, their applicability is limited to situations in which a high degree of multi-modality is not present. Unfortunately, most realistic optimization applications involve design spaces that are clustered with a multitude of local minima. For this reason, a variety of non-gradient-based and heuristic optimization algorithms have been in development over the last 30 years, which are more suitable for a wide range of real-world scale engineering optimization applications.

Many of these algorithms, such as genetic algorithms (GA) [10] and simulated annealing (SA) [14], are patterned or modeled after real-world phenomena. Of the two examples listed for example, the former emulates biological selection, and the latter emulates the physical annealing process of a material solid. Furthermore, these methods rely heavily on random decisions and probability, and it is this feature which gives these

methods their hill-climbing behavior, and allows them to escape local minima in their ultimate search for the global minimum solution. In other words, these methods attempt to temper exploitative searches with some degree of exploration. Much like Tabu search, the searches are allowed to not only intensify in regions of the design space which are found to be advantageous, but they are also allowed to diversify into new regions of the design space [8].

Motivation

The inherent disadvantage of most heuristic optimization algorithms is the poor performance that stems from their inherent random/probabilistic nature. Exploratory behavior is necessary to escape local minima, but often leads the search into undesirable sectors of the design space which might not be easily retreated. For the present research, the ultimate objective is to feed off the strengths of these methods and develop either a brand new standalone heuristic optimization algorithm, or alternatively a meta-heuristic that might help to improve the existing bank of these types of probabilistic based optimization algorithms.

The underlying motivation of this research effort is to eventually apply the methodologies developed in situations where gradient-based optimization algorithms have been found to fail – large scale, multi-modal, multidisciplinary design optimization. In a recent research effort [12], the author found that gradient-based AAO works well in situations where problem size and complexity are small, but not so well for more realistic design problems. In a slightly earlier research effort [2], an AAO implementation displayed very promising results that were seen on small (3 subsystem) linear test systems, with a priori known solutions. It is hoped that the application of the Monty Hall heuristics, the details of which are discussed in the next section, can show improved performance for larger scale, higher complexity AAO applications.

Method Implementation

The Monty Hall problem [16,7,3] is one of the most well-known problems in statistics and probability. The author's interest in the problem is, in great part, what inspired the present research effort. The Monty Hall problem is stated in words as follows:

"You are a participant on Let's Make a Deal. Monty Hall shows you three closed doors. He tells you that two of the closed doors have a goat behind them and that one of the doors has a new car behind it. You pick one door, but before you open it, Monty opens one of the two remaining doors and shows that it hides a goat. He then offers you a chance to switch doors with the remaining closed door. Is it to your advantage to do so?"

The solution to this problem has been wildly debated among mathematicians world-wide for years. The reason for the debate, and the heated difference in opinion, boils down to what, if anything, can be assumed regarding Monty Hall's motivation for opening a curtain. Three distinct possibilities exist: 1. He randomly opens doors. 2. He always opens the door he knows contains nothing. 3. He only opens a door when the contestant has picked the grand prize.

Seemingly, the most reasonable scenario is possibility #2, which, if assumed, yields the following solution to the problem, again stated in words:

"You know that Monty Hall isn't going to show you the curtain with the car behind it. Therefore, by which curtain he opens, he gives you information about the problem. At the start, the car has an equal chance of being behind each curtain. The following three situations have equal probability:

	Curtain 1	*Curtain 2*	*Curtain 3*
Scenario A	*Car*	*Goat*	*Goat*
Scenario B	*Goat*	*Car*	*Goat*
Scenario C	*Goat*	*Goat*	*Car*

Table 1: 3 scenarios in the Monty Hall problem

The probability that your initial guess is correct is therefore 1/3. Sticking to this guess keeps the probability 1/3. Now, let's see what happens supposing that you switch your guess. There are three situations, A, B, and C, with equal probability. Let's say that your original guess was curtain #1. It doesn't matter which one you pick first; it's all the same. If A is true, then Monty Hall will open up curtain #2, or curtain #3. Assuming you switch, you'll guess the other one, and you'll get a goat. That's a loss. If B is true, then Monty Hall must open curtain #3, because he won't open one with a car. If you switch, you'll guess curtain #2, and you'll get a car. That's a win. If C is true, then Monty Hall must likewise open curtain #2. If you switch, you'll get the car. Win again. You have three situations, with equal probability. Switching will give you a loss on one of them, and a win on two. Therefore the probability of a win if you switch is 2/3. Since this is better than the probability of your initial guess of 1/3, it's better to switch."

In the present research, the Monty Hall (MH) problem will be paralleled into an analogous heuristic optimization algorithm, also somewhat inspired by the spirit of the SA, Tabu, and GA approaches. Here, one

design variable will be perturbed on each iteration of the process. In the proposed approach, the MH problem has been generalized from a three-curtain arrangement to an "n" curtain arrangement. In the initial implementation, only 3 curtains have been used, as was the case on the TV show. Behind each curtain lies one of the "p" design variables in the design. If there are more curtains than design variables, then duplicate instances of the design variables are placed behind each curtain. If there are more design variables than curtains, then "n" of the "p" design variables are randomly chosen, and one each is placed behind the available curtains. Here, the location of the "prize" is set to be the design variable that is calculated to be the most critical to the design. (Note: the "prize" design variable is only placed behind a single curtain – duplicate instances of this variable are not permitted to exist, in situations where p > n). This determination is made through the use of sensitivities and effectiveness coefficients [9].

Effectiveness coefficients quantify the impact of a particular design variable X_i on the design at a given point in the design space, and can be defined as:

$$e_{ij} = \frac{\frac{dg_j}{dX_i}}{\frac{dF}{dX_i}} \qquad [1]$$

Here, F is the objective function, and g_j, j=1,m are the inequality constraints. Because there will likely be more than one constraint present, there is a need to assess a measure of sensitivity of a single, cumulative constraint function, C, to each given design variable. This measure C is formed by means of the K.S. function [15], as follows:

$$C = \frac{1}{\rho}\ln\left[\sum_{j=1}^{m}\exp(\rho \cdot g_j)\right] \qquad [2]$$

Note that the value of ρ must be chosen carefully. A smaller value of ρ allows more constraints to participate in the constraint representation, while a larger value of ρ allows the most critical constraints to dominate the constraint representation. Clearly, one can expect some methodologicial dependency on the choice of this parameter. Once this cumulative constraint measure is formulated, a single measure of effectiveness can be formed for each design variable to the cumulative constraint and the objective function:

$$e_i = \frac{\frac{dC}{dX_i}}{\frac{dF}{dX_i}} \qquad [3]$$

The default setting is such that the "prize" variable is assigned to be the variable with minimum computed effectiveness. The user has the option of assigning the "prize" variable to be the variable with maximum effectiveness, or alternatively, the maximum absolute value effectiveness, either of which might introduce more exploratory traits into the search.

The player's "guess" is simulated by a random choice among the number of curtains present, behind which lies one of the system design variables. Next, Monty Hall's "revealed curtain" is simulated by yet another random choice among the remaining curtains – in other words, this selection will not be the "prize" curtain, nor will it be the initially "player chosen" curtain. Finally, the player is given the option to switch from his initial choice, to a different curtain, based on the information conveyed by Monty's revealed curtain. This process is simulated by way of a user-defined "switch percentage", which can range from 0 (NEVER switch) to 1 (ALWAYS switch). In the author's estimation, this behavior can add a bit of "unplanned randomness" to the process, much in the spirit of the Mutation operator from GA optimization.

Finally, the curtain is opened, revealing the variable that is to be perturbed on the current design cycle. For constrained optimization problems, a pseudo-objective function is formed with the following general form:

$$F_{final} = F + r_p * C \qquad [4]$$

Here, r_p is the penalty parameter for the cumulative constraint function; this can be a static or a dynamic algorithm parameter. Once the variable to be altered is chosen, the uni-variate search can proceed. The user has 3 possible choices at this juncture: a. random perturbation of the variable within the variable bounds, b. steepest descent (using analytical or numerical sensitivity) with a random step size, or c. a golden section search.

The acceptance criterion for the proposed move is also user-defined. The user can define a "minimum improvement" threshold that must be overcome for a trial design point to be accepted as the current best. Alternatively, the user can dictate that any improved points may be accepted, or like SA, some degree of "hill-climbing" can be tolerated through the use of a probabilistic acceptance criterion such as that commonly used in a standard SA. Yet another "SA-like" acceptance criterion that the user may choose is that whenever the user chooses the "prize" variable during the decision process, that move is automatically accepted, regardless of the objective function improvement (or lack thereof).

In many realistic optimization problems, a large portion of the design variables will be at their upper or lower bound at a local or at the global optimum. Anticipating this, the MH algorithm checks to see whether a given design variable is at its bound, or near its bound within some small tolerance, before choosing the "prize" variable. Variables that are at their upper or lower bound cannot be chosen to be the "prize" variable, even if the effectiveness calculation dictates such. This should promote a greater degree of exploration in the design space, all the while successively exploiting the most effective design variables.

Once selected, the new trial point is evaluated, and compared to the current best. Then, the cycle counter is decremented, and the process repeats until the user-defined "min iteration" threshold is met or until known convergence is attained. In most cases, the global optimum will not be known to the user. If it is, then the stopping criterion can be set to be some small tolerance away from the known solution.

Complex System Simulation Details

The testing of newly-developed optimization methods requires the use of a wide variety of stable test systems. For this reason, the CASCADE (Complex Application Simulator for the Creation of Analytical Design Equations) simulator has been used in this research for the generation of analytical, multidisciplinary test systems. A thorough description of CASCADE can be found in past literature [11], and a brief overview will be presented here for completeness.

CASCADE is a computer tool that generates a coupled system representation that consists of analytical equations of user-specified size, and has the capability of generating equations that represent both a coupled system analysis and its associated optimization problem. The *structure* of the system is user-dictated; the *semantics* are randomly generated. The analysis portion consists of a band of nonlinear equations that attempt to represent the structure of the coupled nature of the subsystem outputs (behavior variables). Each behavior variable is a polynomial function of both independent inputs (design variables) and dependent inputs (other behavior variables). In this research effort, two different fully-coupled test systems are generated by CASCADE.

Results and Discussion

{Note that the results that follow assume that the reader has an appropriate background in simulated annealing, and the basics of traditional zero-order/gradient based optimization methods. For basics on the former, refer to [14]; for the latter, refer to [19].}

Problem #1

As of this writing, the proposed Monty Hall (*MH*) algorithm is in its infancy stages. Thus far, the algorithm has been tested on numerous problems, the first of which is the famous 2 variable unconstrained "banana" function, shown as follows:

Minimize:

$$F(\mathbf{X}) = 10X_1^4 - 20X_1^2X_2 + 10X_2^2 + X_1^2 - 2X_1 + 5 \qquad [5]$$

The solution of this problem is $X=[1\ \ 1]^T$, which yields an objective function value of $F(\mathbf{X}) = 4$. The design space is plotted in Figure 3.

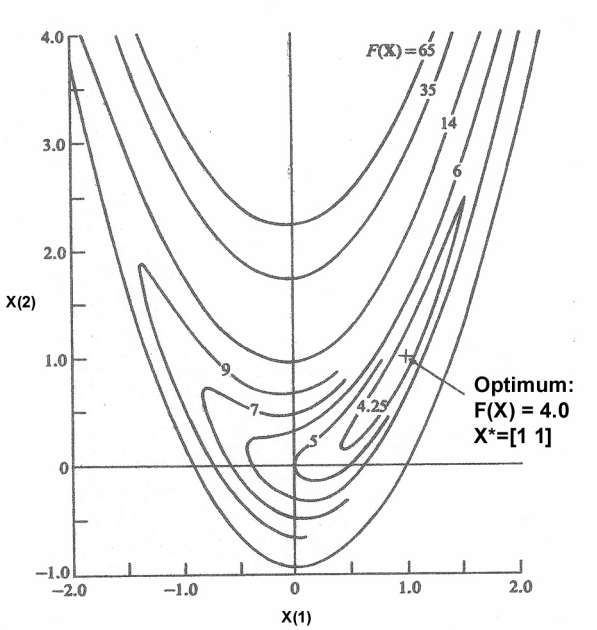

Figure 3: Design space for Test Problem #1

The MH algorithm has been implemented on this test problem, and has been compared to a simple random search (*RS*) method, a gradient-based (*GB*) optimizer by way of MS Excel's internal solver, and a baseline simulated annealing (*SA*) code. One would hope that for such a simple problem, the new MH algorithm would prove itself to perform substantially better than the non-intelligent random search, and at least comparably to the slightly more intelligent simulated annealing search. Preliminary results are indeed promising, and are summarized in Appendix I. For this simulation, the upper and lower bounds have been assigned to be ± 25.0. The simulation has been set to execute for 100000 evaluations, or when the best-found objective function is found to be within $\varepsilon_f = 0.01$ of the known optimal solution. Baseline methodological

settings for each of the four optimization methods are summarized in Table 2.

Method	***Setting***
1. RS	Move limit: *100%*
2. GB	Linear model assumed? *No*
	Variable scaling used? *No*
	Finite difference derivatives: *Central*
	Search technique: *second order (Newton)*
	1-D search estimates: *Quadratic*
3. MH	Acceptance criterion: *improved designs only*
	Uni-variate search type: *golden section*
	Effectiveness: *max (dF/dX)*
	Number of curtains: *3*
	"Switch" percentage: *100%*
4. SA	Move limit: *40%*
	Initial temperature: *22000*
	Final temperature: *1.0*
	Decrement rule: *0.99*
	Iterations at each T to reach steady-state: *100*

Table 2: baseline optimization settings

A total of 10 trial executions were performed for each method. Clearly, each of the 4 optimization methods has its strengths and weaknesses. The Gradient-based solver attains the global optimum solution of 4.0 from 7 of the 10 starting points, and does so very quickly – in 10 or fewer optimization iterations. Due to the high non-linearity of the design space, as seen in Figure 3, the gradient-based solver solution path becomes trapped in local minima for the other 3 trial runs. The random search attained vastly improved designs for all 10 trial runs, and found near optimal solutions in trial runs 1,7, and 10. Note however the tremendous expense in attaining these, and the other 7 inferior final design points – 100000 total function evaluations. Similar, but slightly improved results were attained using the simulated annealing optimizer. A near optimum solution was found on trial run 4, and consistently "close to optimum" solutions were found on most of the other 9 trial runs, but again, at the expense of many function evaluations.

On this first test problem, the Monty Hall algorithm shows definite promise. The exact optimum solution (within the stated tolerance) was attained on all 10 trial runs. The function evaluation counts were found to be reasonable as well – 2 of the 10 runs had fewer than 100 cycle iterations, and all runs converged in fewer than 5000 iterations. Though more iterations were required than that required for the gradient-based optimizer, global optimality was attained from ***every*** starting point. The end user may wish to sacrifice some degree of efficiency for "guaranteed" performance. Figures 4 and 5 summarize the performance and efficiency-related findings shown in Appendix I.

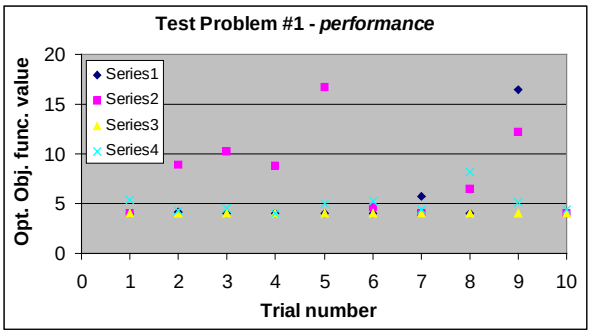

Figure 4: Problem #1 performance

Figure 5: Problem #1 efficiency

In these and all result plots that follow, the 4 series are notes as follows: ***Series 1 (diamond) – GB, Series 2 (square) – RS, Series 3 (triangle) – MH, Series 4 (cross) – SA.***

Problem 2:

The second test problem is a bit more realistic in size and scope. It is a multidisciplinary test problem created by the CASCADE simulator, and has 3 subsystems, each of which has 1 output behavior variable, and 1 design variable. There are 2 inequality constraints associated with each subsystem, making a total of 6 constraints for the problem. There is one system-level objective function, which is to be minimized. The semantics have been omitted for simplicity; nonetheless, the structure of test problem 2 is denoted as shown in equation [6] and Figure 6:

$$
\begin{aligned}
&\textbf{\textit{Minimize}}: && F(X_1, X_2, X_3, Y_1, Y_2, Y_3) \\
&\textbf{\textit{Subject to}}: && g_1(X_1, X_2, X_3, Y_1, Y_2, Y_3) \le 0 \\
& && g_2(X_1, X_2, X_3, Y_1, Y_2, Y_3) \le 0 \\
& && g_3(X_1, X_2, X_3, Y_1, Y_2, Y_3) \le 0 \\
& && g_4(X_1, X_2, X_3, Y_1, Y_2, Y_3) \le 0 \\
& && g_5(X_1, X_2, X_3, Y_1, Y_2, Y_3) \le 0 \\
& && g_6(X_1, X_2, X_3, Y_1, Y_2, Y_3) \le 0 \\
& && -9999 \le X_1, X_2, X_3 \le 9999
\end{aligned}
\qquad [6]
$$

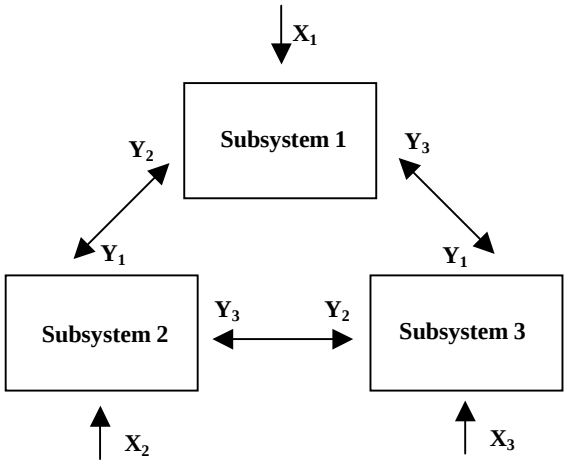

Figure 6: Structure for Test Problem #2

The objective and constraints are non-linear, highly non-convex polynomial functions of the design variables X and the behavior (coupling) variables Y. In a true MDO problem, there would also be non-linear state equations associated with each of the behavior variables Y. In the corresponding AAO problem formulation, each of these equations would necessarily result in an additional equality constraint. For the sake of algorithmic development, these equations have been omitted, and the process temporarily simplified by treating the behavior variables as constant values, obtained using an off-line fixed-point iteration procedure [13] using the initialized values of the design vector **X**.

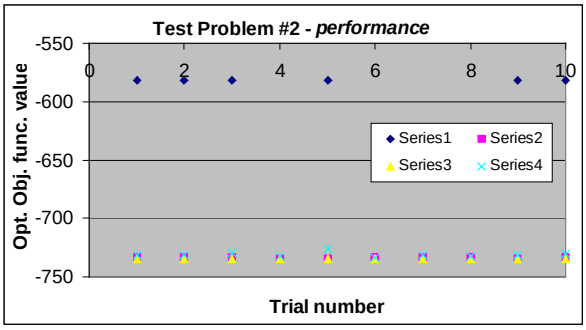

Figure 7: Problem #2 performance

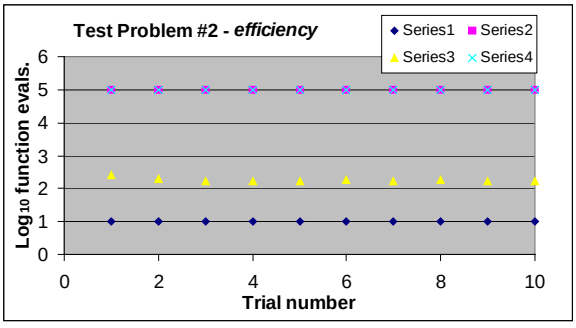

Figure 8: Problem #2 efficiency

Numerous observations can be made regarding the results of this test problem, which are summarized in Appendix II. The gradient-based solver attained what I speculate is the global optimum solution of –734.24 on 4 of its 10 trial runs. At this optimum, only the 6th inequality constraint is active; the other 5 constraints are satisfied but not active. For the other 6 trial runs, the gradient-based solver attained vastly improved solutions, but converged at a very stingy local optimum (of value –581.55). At this design point, 4 of the 6 inequality constraints are active; the other 2 are satisfied. Note however that all 10 trial runs attained solutions in 10 or fewer iterations – this is very efficient solution improvement. This problem does a nice job of demonstrating that a gradient-based optimizer can fail to attain global optimality in design spaces that are highly volatile. The random search and simulated annealing optimizers both do a good job of attaining very near optimal solutions – consistently more so than the gradient-based solver. Clearly, the drawback is the computational expense – 100000 iterations for a full-iteration exploration, and a full computational anneal, respectively. Once again, the MH algorithm proves itself to be a solid optimization alternative. In this problem, the golden section uni-variate search was once again used, but this time, due to the presence of 6 non-linear constraints (and after initial testing difficulties), one of the two bounds on the optimizer had to be randomized on each optimization cycle, between 0 and ± bound. Once this bounding strategy was employed, the suspected global optimum is converged upon during all 10 trial runs. Once again, the iteration count is also modest, being in the "mid hundreds" for all runs. This is not nearly as efficient as the "ten or fewer" efficiency of the gradient-based solver, but again, this might be a small price to pay for "guaranteed" global optimality. Figures 7 and 8 summarize the performance and efficiency-related findings shown in Appendix II.

Problem 3:

The third test problem is a much larger and more realistic multidisciplinary test problem, again created by the CASCADE simulator. Here, there are 100 subsystems, 1 design variable and 1 behavior variable per subsystem. There are a total of 10 inequality constraints for the problem. There is one system-level objective function, which is again to be minimized. The semantics have again been omitted for simplicity;

nonetheless, the structure of test problem 3 is denoted as shown in equation [7] and Figure 9:

Minimize:
$F(X_1, X_2, X_3, \ldots X_{100}, Y_1, Y_2, Y_3, \ldots Y_{100})$

Subject to:
$g_1(X_1, X_2, X_3, \ldots X_{100}, Y_1, Y_2, Y_3, \ldots Y_{100}) \le 0$
$g_2(X_1, X_2, X_3, \ldots X_{100}, Y_1, Y_2, Y_3, \ldots Y_{100}) \le 0$
$g_3(X_1, X_2, X_3, \ldots X_{100}, Y_1, Y_2, Y_3, \ldots Y_{100}) \le 0$
$g_4(X_1, X_2, X_3, \ldots X_{100}, Y_1, Y_2, Y_3, \ldots Y_{100}) \le 0$
$g_5(X_1, X_2, X_3, \ldots X_{100}, Y_1, Y_2, Y_3, \ldots Y_{100}) \le 0$
$g_6(X_1, X_2, X_3, \ldots X_{100}, Y_1, Y_2, Y_3, \ldots Y_{100}) \le 0$
$g_7(X_1, X_2, X_3, \ldots X_{100}, Y_1, Y_2, Y_3, \ldots Y_{100}) \le 0$
$g_8(X_1, X_2, X_3, \ldots X_{100}, Y_1, Y_2, Y_3, \ldots Y_{100}) \le 0$
$g_9(X_1, X_2, X_3, \ldots X_{100}, Y_1, Y_2, Y_3, \ldots Y_{100}) \le 0$
$g_{10}(X_1, X_2, X_3, \ldots X_{100}, Y_1, Y_2, Y_3, \ldots Y_{100}) \le 0$
$-9999 \le X_1, X_2, X_3 \ldots X_{100} \le 9999$

[7]

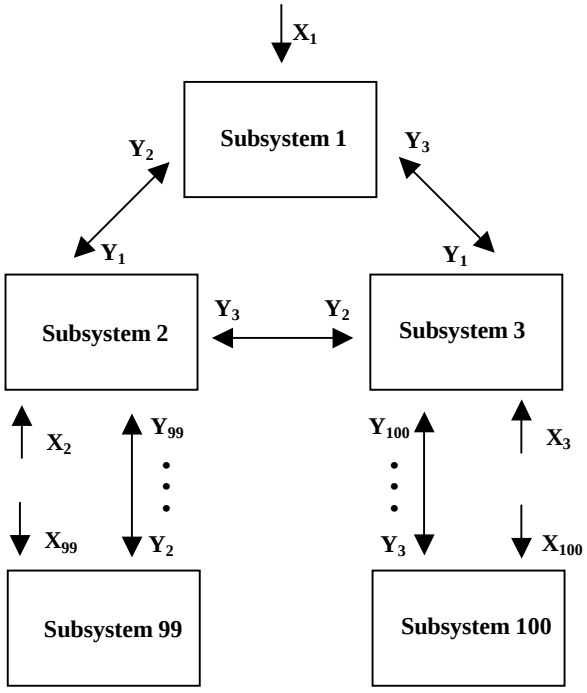

Figure 9: Structure for Test Problem #3

Similar observations can be made regarding this third test systems as those made in the first two, and results are summarized in Appendix III. The gradient-based optimizer attains vastly improved solutions with a "small" cost (100 or fewer optimization iterations). Clearly however, once we execute the other optimizers, we quickly see that the best gradient-based converged solutions are a substantial distance away from numerous other found "best" solutions. This time, fewer iterations were executed for the random search and simulated annealing codes - 10000. The simulated annealing code also had its minimum temperature towered from 1.0 in previous runs to 0.001, for a longer "near zero" anneal. Both method demonstrated consistently better results than the gradient-based searches, with results averaging around the mid – 3600.0 range, but again, at a substantially higher computational cost. Again, the Monty Hall algorithm shows promise. The suspected global optimum is attained for all 10 runs: -3773.9. At this solution, 6 of the 10 design constraints are active, and a vast majority of the design variables converge to their upper or lower bound. Note that this solution **was not attained** by any of the other 3 algorithms on any of the trial runs. Once again, the iteration count was found to be "reasonable" for the MH algorithm (though decidedly greater than the corresponding count for gradient-based optimization), averaging around 1000 iterations per trial run. In this case, this is the price paid for vastly improved (and possible global optimum) performance. Figures 10 and 11 summarize the performance and efficiency-related findings shown in Appendix III.

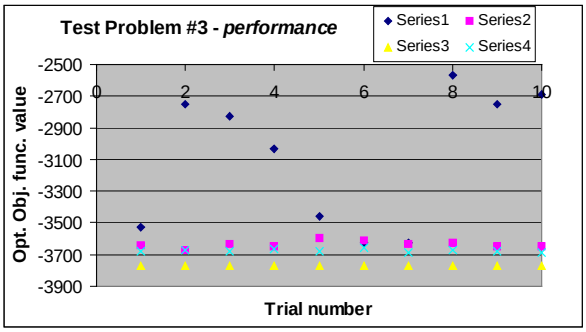

Figure 10: Problem #3 performance

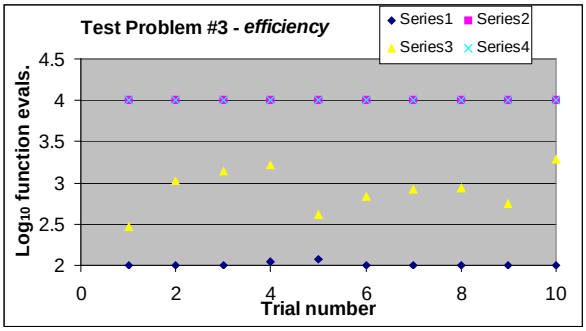

Figure 11: Problem #3 efficiency

Conclusions and Future Work

This paper has presented preliminary findings of the development of a new heuristic optimization algorithm, inspired by the famous "Monty Hall" problem. Similar to numerous existing non-traditional optimization algorithms that is modeled after (such as simulated annealing and tabu search), the Monty Hall algorithm attempts to conduct an exploitative search while simultaneously in pursuit of global optimality by way of a large degree of exploratory search behavior as well. The motivation for the development of this algorithm is to find more successful alternatives to solving large scale multidisciplinary design problems using an all-at-once optimization approach.

In this paper, 3 test problems were examined: the famous 2-variable unconstrained "banana" problem, and 2 constrained, multidisciplinary CASCADE-generated simulations: one having 3 design variables, 3 behavior variables, and 6 constraints, and the second having 100 design variables, 100 behavior variables, and 10 constraints. Furthermore, for purposes of comparison, 3 other optimization methods were employed: gradient-based optimization by way of MS Excel's solver, a pure random search, and an "intelligent random" search in the form of simulated annealing. Preliminary findings have been very promising. General observed trends are summarized as follows. Excel's internal gradient-based solver consistently attained design improvement; often times global optimality for the smaller two problems. Most important to note is that it did so very quickly, as would be expected with its inherent second-order calculus-based intelligence. The problem using this gradient-based solver is reliability – unless the starting point is such that the global optimum is bounded, the solution path will likely converge to a local minimum. This happened frequently, especially in the case of the 100 variable problem. At the other extreme lies the random search, which consistently attained vastly improved, often near-global optima for all 3 test problems. Clearly, the problem that was observed with this approach is one of efficiency – thousands or millions of iterations are typically required to find near-optimum solutions in any optimization problem of realistic size and scope. Much of the same can be said for simulated annealing, which, with appropriate settings, and after a full anneal (i.e. *many* iterations) often leads to near-optimum solutions that can be superior to those attained by a conventional random search. The Monty Hall algorithm attempted to meet the best of both worlds – global or near-global optimality attained in a "reasonable" number of total optimization cycles. For all 3 problems, and for all 10 test runs for each, the suspected global optimum solution was attained, and always with far fewer iterations required that that for GB or SA optimization. This number of iterations was typically greater than that required for GB optimization convergence, but seemingly with a much higher level of confidence that a superior final minima will be attained.

Having explored this topic enough to see that there is enough potential to explore this research further, the author has numerous ideas for future work. First and foremost, the algorithm must be put to the test on larger and more complex problems. Particularly, these problems should be "true" multidisciplinary in nature, with analysis variables (and the AAO equality constraints that result) included in the final problem formulation. Now that the author has a degree of confidence that the algorithm can produce desirable results for challenging problems, it will be beneficial to return to the very same multidisciplinary test problems which demonstrated degrading performance in the AAO solution strategy with increased problem size (Hulme and Bloebaum, 1998). All of these test problems utilized a gradient-based optimizer exclusively; the author speculates that the hybrid heuristic/derivative-based approaches inherent in the MH algorithm will result in improved performance for AAO multidisciplinary optimization.

Many of the inherent dynamics in the overall "Monty Hall" decision process have thus far gone under-explored. Namely, the author looks forward to investigate the impact (if any) on varying the "user switch" percentage on a given problem. Remember that in a 3 curtain arrangement, remaining with your initial choice gives the user a 1/3 chance of choosing the prize, while switching ones choice gives the user a 2/3 chance of winning. Does this decision problem have any impact on the Monty Hall optimization analog? This question begs another – what effects (if any) would a larger curtain arrangement have? In this research, 3 curtains were used, as was the case on the TV show. It might make more sense (to avoid the "underrepresented variable" scenario that arises on any problem with more than 3 design variables) to exactly match the number of curtains to the number of total design variables. Clearly however, this means that the percentage of times that the "prize" (highest effectiveness) variable is chosen will necessarily decrease, resulting most likely in a far more exploratory, and less efficient search. These however, are the decisions that are likely ver problem and situation dependent, and are best left to the discretion of the end user(s) solving the problem at hand.

Acknowledgement

The author would like to acknowledge the partial funding support of Sheldon Silver and the New York State Assembly for their support of NYSCEDII, the research organization for which the present work was performed.

References

1 Balling, R. J., and Sobieszczanski-Sobieski, J., "Optimization of Coupled Systems: A Critical Overview of Approaches." AIAA Paper 94-4339, September, 1994.

2 Balling, R.J., and Wilkinson, C.A., "Execution of Multidisciplinary Design Optimization Approaches on Common Test Problems." AIAA Journal, Vol. 35, No. 1, January, 1997, pp. 178-186.

3 Barbeau, E., "The Problem of the Car and Goats." College Mathematics Journal, Volume 24, Number 2, Mar 1993, page 149.

4 Bloebaum, C.L., Hajela, P., and Sobieszczanski-Sobieski, J., "Non-Hierarchic System Decomposition in Structural Optimization." Third USAF/NASA Symposium on Recent Advances in Multidisciplinary Analysis and Optimization, San Francisco, CA, September, 1990.

5 Cramer, E.J. et al., "On Alternative Problem Formulations for Multidisciplinary Design Optimization." Fourth AIAA /NASA /ISSMO Symposium on Multidisciplinary Analysis and Optimization, Cleveland, OH, September, 1992.

6 Cramer, E.J. et al., "Problem Formulation for Multidisciplinary Optimization." SIAM Journal of Optimization, No. 4, pp. 754-776, November, 1994.

7 Gillman, L., "The Car and the Goats." American Mathematical Monthly, Volume 99, Number 1, Jan 1992, page 3.

8 Glover, F., "Tabu Search, Part I." ORSA Journal on Computing 1:3, 1989, pages: 190-206.

9 Hajela, P., Sobieszczanski-Sobieski, J., "The Controlled Growth Method – A Tool for Structural Optimization", Proceedings of the AIAA/ASME/ASCE/AHS 22nd Structures, Structural Dynamics, and Materials Conference, 1981.

10 Holland J.H., "Adaptation in Natural and Artificial System." Ann Arbor, The University of Michigan Press, 1975.

11 Hulme, K.F., and Bloebaum, C.L., "Development of a Multidisciplinary Design Optimization Test Simulator." Structural Optimization, Volume 14, Number 2-3, October, 1997.

12 Hulme, K.F., and Bloebaum, C.L., "A Comparison of Solution Strategies for Simulation-based Multidisciplinary Design Optimization." Seventh AIAA/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, St. Louis, MO, September, 1998, pp. 2143-2153.

13 Istratescu, V.I., "Fixed Point Theory. An Introduction." Math. and Its Applications, D. Reidel Publishing Co., Dordrecht, 1981.

14 Kirkpatrick, S., Gelatt, C., and Vecchi, M., "Optimization by Simulated Annealing." Science, 220 (1983) pp. 671-680.

15 Kreisselmeier, G. and Steinhauser, R., "Systematic Control Design by Optimizing a Vector Performance Index." International Federation of Active Control Symposium on Computer-Aided Design of Control Systems, Zurich, Switzerland, August, 1979.

16 Selvin, S., "A Problem in Probability." American Statistician, Volume 29, Number 1, February 1975, page 67.

17 Sobieszczanski-Sobieski, J., "A Linear Decomposition Method for Optimization Problems - Blueprint for Development." NASA Technical Memorandum 83248, 1982.

18 Sobieszczanski-Sobieski, J., "The Sensitivity of Complex, Internally Coupled Systems." AIAA Journal, Volume 28, No. 1, 1990, pp. 153-160.

19 Vanderplaats, G. N., "Numerical Optimization Techniques for Engineering Design: with Applications." McGraw Hill, New York, N.Y., 1984.

Appendix I – Problem #1 result summary

Trial	GB initial F(X)	GB final F(X)	GB Func. evals.	RS initial F(X)	RS final F(X)	RS Func. evals.	MH initial F(X)	MH final F(X)	MH Func. evals.	SA Initial F(X)	SA Final F(X)	SA Func. Evals.
1	207589.	4.037	< 10	1845299.	4.036	100000	2568844.	4.010	3557	267618.	5.414	100000
2	442671.8	4.138	< 10	770792.9	8.901	100000	191677.2	4.010	3528	58536.0	4.241	100000
3	8.1855	4.0	< 100	589534.8	10.282	100000	12495.76	4.010	3203	178718.2	4.567	100000
4	3568.00	4.0	< 100	204851.1	8.739	100000	1291636.	4.010	3489	89.276	4.032	100000
5	47.242	4.0	< 100	4516.76	16.755	100000	630681.2	4.010	154	1598442.	5.042	100000
6	3127325.	4.038	< 10	1181789.	4.651	100000	133395.5	4.010	3574	2536.18	5.242	100000
7	51158.14	5.675	< 10	73854.62	4.051	100000	3047727.	4.010	3669	1261240.	4.377	100000
8	84.812	4.0	< 100	22292.23	6.511	100000	3928390.	4.010	43	331363.8	8.146	100000
9	1785015.	16.489	< 10	3132936.	12.209	100000	58454.9	4.010	3537	674944.7	5.174	100000
10	190609.7	4.073	< 10	964.411	4.039	100000	53628.8	4.010	44	122.43	4.369	100000

Appendix II – Problem #2 result summary

Trial	GB initial F(X)	GB final F(X)	GB Func. evals.	RS initial F(X)	RS final F(X)	RS Func. evals.	MH initial F(X)	MH final F(X)	MH Func. evals.	SA Initial F(X)	SA Final F(X)	SA Func. Evals.
1	-6.72	-581.55	< 10	-159.32	-734.10	100000	229.38	-734.24	255	33.54	-731.45	100000
2	170.27	-581.55	< 10	-198.35	-734.00	100000	-347.48	-734.24	197	101.21	-731.20	100000
3	-511.73	-581.55	< 10	209.68	-733.99	100000	-9.70	-734.24	174	201.61	-729.38	100000
4	-229.96	-734.24	< 10	202.99	-734.13	100000	206.72	-734.24	175	-25.29	-732.78	100000
5	-579.98	-581.55	< 10	-161.20	-734.11	100000	298.00	-734.24	172	-157.90	-726.31	100000
6	-119.20	-734.24	< 10	-42.08	-733.91	100000	314.87	-734.24	182	245.19	-733.37	100000
7	58.32	-734.24	< 10	171.00	-734.02	100000	-702.56	-734.24	174	113.77	-731.56	100000
8	-168.83	-734.24	< 10	108.85	-734.08	100000	29.06	-734.24	180	-180.54	-733.03	100000
9	141.27	-581.55	< 10	19.45	-734.14	100000	-138.44	-734.24	169	-181.02	-731.80	100000
10	-480.39	-581.55	< 10	-70.63	-733.82	100000	-57.99	-734.24	166	-179.95	-730.10	100000

Appendix III – Problem #3 result summary

Trial	GB initial F(X)	GB final F(X)	GB Func. evals.	RS initial F(X)	RS final F(X)	RS Func. evals.	MH initial F(X)	MH final F(X)	MH Func. evals.	SA Initial F(X)	SA Final F(X)	SA Func. Evals.
1	526.5	-3530.2	< 100	561.63	-3637.7	10000	-330.11	-3773.9	295	528.32	-3677.0	10000
2	-176.2	-2748.1	< 100	562.76	-3673.4	10000	-108.82	-3773.9	1069	564.51	-3671.1	10000
3	-25.72	-2830.7	< 100	529.29	-3632.3	10000	-527.29	-3773.9	1403	527.78	-3679.2	10000
4	-31.24	-3035.3	< 110	569.06	-3647.4	10000	149.02	-3773.9	1658	520.72	-3662.8	10000
5	171.9	-3461.6	< 120	527.21	-3595.9	10000	278.92	-3773.9	406	529.34	-3678.9	10000
6	-55.63	-3628.4	< 100	580.97	-3610.2	10000	-332.87	-3773.9	681	557.28	-3659.8	10000
7	-135.20	-3626.7	< 100	496.39	-3635.5	10000	-262.08	-3773.9	823	470.83	-3684.3	10000
8	209.21	-2569.8	< 100	466.17	-3624.9	10000	20.24	-3773.9	853	518.71	-3670.8	10000
9	109.16	-2751.7	< 100	525.86	-3646.5	10000	-169.84	-3773.9	550	587.00	-3678.8	10000
10	-351.33	-2688.0	< 100	555.68	-3646.4	10000	285.64	-3773.9	1962	527.01	-3683.6	10000