Proceedings of the 2004 Fall Simulation Interoperability Workshop, Orlando, FL, September 2004.

Coercion through Optimization: A Classification of Optimization Techniques

Sarah Waziruddin
David C. Brogan
Paul F. Reynolds Jr.

Modeling and Simulation Technology Research Initiative
University of Virginia
151 Engineer's Way, P.O.Box 400740
Charlottesville, Virginia, 22904-4740
(434) 982-2296, (434) 982-2211, (434) 924-1039
{swaziruddin | dbrogan | reynolds}@cs.virginia.edu

Keywords: reuse, optimization, coercing simulations

ABSTRACT: *Optimization techniques have been used to search for optimal values for decision variables and input variables associated with a simulation. More recently we have explored a mixed-method approach, mixing optimization and code modification, for "coercing" simulations to meet new requirements. The coercion process, P, transforms a simulation that meets requirement R to meet a new requirement, R', without necessarily resorting to redesign or reimplementation. Coercion is a semi-automated process because it combines optimizations and code modifications. A coercion can be characterized as a regular expression P = [m*, o*]* where m and o represent code modification and optimization respectively. P can consist of any number, including zero, of modifications and/or optimizations in any order. The use of optimization is encouraged because it increases the level of automation in the coercion process.*

We explore the question of determining best optimization techniques for coercion. We begin by considering classifications of optimization techniques already existing in the simulation community and apply these in the context of coercion. We consider issues such as computation time, set-up time, and avoidance of local minima that contribute towards the ease of use of each optimization technique. We discuss insights we expect to gain from the use of optimization and identify tools necessary to gain these insights. Additionally, we discuss how the result of one optimization may affect the outcome of another in a coercion sequence, P. We expect our results to serve as a guide to optimization method selection for simulation practitioners who wish to employ coercion.

1. Introduction

Simulation transformation occurs frequently, primarily to satisfy the desire for code reuse and evolving requirements. It is in the best interest of the simulation community to develop the best methods possible for conducting transformations correctly, efficiently, and with a minimum of stress for the people involved. We explore novel uses of optimization in support of a broader transformation strategy in this paper.

Transforming a simulation to meet a new requirement typically requires extensive amounts of code modification. We have proposed an alternative, coercion, which was introduced by Reynolds [1] and refined by Waziruddin et al. [2]. Coercion is a multi-step process that combines modification and optimization to transform a simulation to meet an achievable requirement it was not originally designed to meet.

The coercion process, P, is characterized by a sequence of optimizations and code modifications occurring in any order. P is defined by the regular expression:

P = [*m**, *o**]*

where *m* and *o* represent code modification and optimization respectively. Thus, sequences such as "oommomomoo" and "mmom" are possible, representing sequences containing modification and optimization of length ten and four respectively. The utility of interweaving optimization and modification has been argued in prior work [2].

The process of transforming a simulation to satisfy a new requirement consists of the simulationist choosing optimization or modification at any given step. The coercion process P represents the steps taken to transform a simulation S_0 exhibiting behavior B_0 satisfying requirement R, to a simulation S_n exhibiting behavior B_n satisfying requirement R'. The

simulationist determines the culmination of the coercion process by subjectively assessing how well B_n satisfies requirement R'.

Note, an infinite number of final simulation instances S_n can be produced; each instance resulting from an infinite number of coercion paths. For example, code modification could always be used to transform S_0 to meet R' or the simulationist may construct a sequence of activities, say characterized by "omo," that also leads to meeting R'. The infinite number of instances of S_n can produce an infinite number of behaviors B_n which can all satisfy the target requirement R'. The simulationist's goal is to determine the best, lowest effort, sequence in the coercion process to find an S_n and B_n to satisfy the requirement R'. Determining the optimal sequence is beyond the reaches of current technology. However, an exciting alternative is to identify procedures that increase the likelihood that a best, or near best, sequence of actions will be chosen. We address the identification of these procedures in this paper.

Recent work extends the notion of coercion introduced by Reynolds [1]. Drewry et al. [3] demonstrate numerical optimization as an effective technique to coerce an environmental simulation, DOLY, to meet requirements established by the output of a higher resolution simulation, CANOAK. Optimal values for a selected set of parameters, which were originally constants, in DOLY are computed using an automated optimization process. The authors execute a coercion process that consists of a sequence of optimizations. This process yields a simulation S_n exhibiting a behavior B_n that satisfies the requirement R' defined by CANOAK.

Carnahan et al. [4] utilize a search technique to coerce a low-resolution particle to produce output similar to that produced by a high-resolution physical simulation of a bicyclist. The authors extend the coercion process further by specifying the roles of simulationist, subject-matter expert, and automation in the coercion process and defining an iterative process whereby a subject-matter expert proposes changes and comparison metrics, a simulationist implements changes, and the subject-matter expert reviews the effects.

These previous papers demonstrate different ways to use optimization in the coercion process. However, they do not attempt to define a set of best practices for the process. Our long-term goal is to discover a way to use coercion such that $E_P(S_0, R') < E_{MOD}(S_0, R')$ where $E_P(S_0, R')$ is the effort required to transform a simulation S_0 using coercion and $E_{MOD}(S_0, R')$ is the effort required to transform simulation S_0 using modification only. This paper's goal is to provide a simulationist with information about various optimization techniques and the tools needed to discover a coercion process, P, such that the effort invested in transforming a simulation S_0 to meet an achievable requirement R' using coercion is much less than the effort required to transform a simulation using only code modification.

We are interested in the role of optimization in the coercion process. Simulation optimization has emerged as an important field in the simulation community and is described as "optimization of performance measures based on output from stochastic (primarily discrete-event) simulations" [5]. Several papers in the literature [6, 7, 8, 9, 10, 11, 12] survey simulation optimization techniques and present various classifications thereof. These traditional monolithic applications of optimization are limited to situations when the number of unknown parameters is small and the behavioral requirements of the simulation can be represented formulaically. In coercion, we further utilize optimization techniques to produce insight about what parameters most influence behavior, what the parameter values should be, and when code modification is needed. In this paper, we classify optimization techniques for their ability to provide valuable insight that reduces the total effort required for successful coercion.

2. Coercion Convergence Property

Coercion has the property of convergence. This property guarantees that any simulation undergoing transformation using coercion will eventually satisfy R' given R' is achievable. It holds because at any step in coercing S_0 to meet R', S_0 or one of its transformed derivatives S_i, can be modified by writing or changing as much code as needed. Because R' is realizable, S_0 or a subsequent S_i can be modified to meet it.

The convergence property does not preclude occasional divergences on any given step in the coercion process. Optimizations may cause the coercion process to diverge from useful behaviors while code modifications may also produce unexpected, undesirable behaviors. Often, divergence enhances exploration and provides insight to find the best transformation. A simulationist using coercion sometimes must allow the process to diverge from a seemingly effective solution to explore alternative solutions, thus gaining insight about the simulation. Motivated by the benefits of divergence in the coercion process, we examine semi-automated optimization methods in which exploration is encouraged.

2.1 The Coercion Invariant

Coercion is a multi-step process consisting of convergence and occasional divergences. We seek a goal to meet at each step in the coercion process. A coercion invariant is a set of conditions that hold true throughout the coercion process and it is intended to be similar to a loop invariant [13], where as long as programmer-specified conditions are not violated, the loop will function as the programmer intended.

To contribute to the specification of a coercion invariant, we define a function $D(S_i, I_i)$ that measures the amount of effort required to complete the transformation of simulation S_i given insight I_i. We assume, after the i^{th} step of the coercion process, $D(S_i, I_i)$ is less than $D(S_{i-1}, I_{i-1})$ either because S_i requires less effort to satisfy R' and/or the insight gained in this step will provide future assistance in transforming S_i. Note, it is not necessary to gain insight about an S_i and transform it closer to satisfying R' at each step i. Divergence from satisfying R' is acceptable because while the simulation, Si, is moving away from satisfying R', the simulationist is gaining valuable insight. Conversely, it is not necessary to gain insight at every step of the coercion process as long as S_i is progressing towards meeting R'.

Given our assumption, anywhere in the coercion process where insight is gained, the benefits of that insight outweigh the effort expended to acquire it. More formally, if $effort_i$ is the effort expended on step i, $D(S_{i-1}, I_{i-1}) - D(S_i, I_i) > effort_i$. Because every step of coercion involves a non-zero amount of effort, $D(S_i, I_i) < D(S_{i-1}, I_{i-1})$ for all steps i. This inequality states that as i, the number of steps in the coercion process increases, the process requires less remaining effort to identify an S_i that satisfies R'. We leave details of the D function unspecified; noting only that it should exist given our assumption about the benefit of insight. The inequality $D(S_i, I_i) < D(S_{i-1}, I_{i-1})$ becomes our coercion invariant, $\mathbf{C_I}$. Because D(S, I) is a strictly monotonic function, coercion is guaranteed to terminate.

We clarify the role of $\mathbf{C_I}$ in the coercion process by presenting a formalism for coercion.

Pre: (S_0 is a simulation satisfying requirement R) AND (i == 0) AND $\mathbf{C_I}$

WHILE **B**: S_i does not satisfy requirement R'
Do **A**: i++;
(i^{th} step in the coercion process);
END

Post: (S_i satisfies R') AND $\mathbf{C_I}$

Pre is the pre-condition to coercion and consists of three terms. The first term states that S_0, the simulation we wish to transform, satisfies requirement R. The second states that the coercion process starts at zero iterations and the last term states that C_I, the coercion invariant, is valid at the start of the coercion process. The while loop represents the coercion process and states that coercion is carried out step-by-step until S_0 is transformed to some Sn, a simulation that exhibits a behavior Bn that satisfies the requirement, R'.

Post is the post-condition of the coercion process and consists of two terms. The first term states that the simulation produced by the last step in the coercion process is a simulation that satisfies a new requirement R' and the second term states that the coercion invariant, $\mathbf{C_I}$, is valid at the end of the coercion process.

$\mathbf{C_I}$ must satisfy the following four rules:

1. **Pre** => C_I: The coercion invariant must be true initially.
2. { $\mathbf{C_I}$ } **B** { $\mathbf{C_I}$ }: Evaluating B will not change the validity of C_I.
3. { $\mathbf{C_I}$ AND **B**} **A** { $\mathbf{C_I}$ }: The coercion invariant is not changed during the coercion process.
4. ($\mathbf{C_I}$ AND (NOT **B**)) => **Post**: If the coercion invariant is true and B is false, then the post-condition is implied.

As we have noted, the i^{th} coercion step can either be an optimization or a modification. We seek techniques that support the coercion invariant; techniques that permit even a naïve coercer to accomplish $D(S_i, I_i) < D(S_{i-1}, I_{i-1})$. Motivated by the potential to gain insight through the use of optimization during coercion, we pursue the opportunity to use optimization to uphold the coercion invariant. In the next section, we classify optimization techniques by the features that contribute to a reduction in the effort required for simulation transformation. Our planned use for optimization differs significantly from its more monolithic use in the simulation community. The role of optimization in coercion is one that provides insight and transformation capability while optimization is used in the simulation community to converge on one best numerical value. Thus, while the simulation optimization community has classified optimization techniques, our evaluation is conducted in keeping with the needs of coercion and the goal that $D(S_i, I_i) < D(S_{i-1}, I_{i-1})$.

3. Optimization Techniques

We review the basics of optimization techniques briefly before presenting our classification. It is

important that the optimization techniques used with coercion provide opportunities for the optimizer to converge on a value needed for transformation or provide an increase in insight about the simulation. The two elements that most directly affect the success of an optimization technique are the quantity and domain of decision variables and the objective function [6]. Identifying the decision variables and the objective function in an optimization problem often requires familiarity with the available optimization techniques and awareness of how these techniques interface with the system undergoing optimization.

Decision variables are parameters to a simulation that are deemed to affect the output in a significant manner. Selecting the best set of decision variables can sometimes be a challenge because it is difficult to ascertain which variables affect each specific behavior in a simulation. Logic determining control flow can also be classified as a decision variable. Condition statements, for example, can be optimized to produce a transformation or to gain insight about the simulation just as variables within the simulation can. The domain of potential values for decision variables are typically restricted by constraints set by the user.

The objective function serves as a surrogate for the user in the optimization by evaluating the quality of the simulation's behavior during the automatic search for optimal decision variable values. It is imperative that the objective function accurately measure the degree to which the simulation's behavior matches the user's goals. If the user has an idea of how to rate the quality of a candidate solution, then identifying an objective function to use is relatively easy. However, in most cases, users do not know exactly what the objective function should be. This causes users to run multiple optimization techniques, each with a different objective function, with the goal of eventually converging on one that rates the quality of a solution to their satisfaction. Ideally the objective function will not only represent the binary state of being satisfactory, but it will also reveal intermediate degrees of success.

3.1 Criteria to Determine Ease of Use

Traditional uses of optimization focus on discovering the best value for a decision variable for a given usage scenario. This monolithic application of optimization is typically executed once to find the optimal values of the decision variables. When used to accomplish coercion, optimization not only provides optimal values for decision variables, it also provides insight about the quality of the transformation. We therefore broaden the description of the role optimization plays in coercion and define attributes that make an optimization technique easy to use in the coercion context. We evaluate three major types of optimization techniques, heuristic, gradient-based, and response surface methodology. The three criteria we use to determine ease of use are: Computation time, set-up time and avoidance of local minima.

Some optimization techniques are more computationally expensive than others and thus the time required to complete an optimization is an important criterion. When optimization is used as a step in the coercion process, it directly impacts how long one must wait to determine optimal values for the decision variables. However, coercion permits an alternative use of optimization where the optimization can be preempted and intermediate results inspected. If one can interrupt an optimization and gain valuable insight, the overall amount of time spent in the coercion process will be reduced because insight gained from one partial iteration of an optimization can help the user with the set-up of other optimization techniques and/or identifying functionality that needs to change in the simulation. Another component of the computation time criterion is how well the time invested in the optimization translates to the quality of the decision variable values found. If there were no direct correlation between the time invested and the quality of results, then waiting for an optimization technique to converge may not result in improved simulation behavior. However, if the user is more interested in gaining insight about the relationship between decision variables and simulation behavior, then an optimization that takes a long time or fails to converge may provide valuable insight about the simulation.

The set-up time required of an optimization technique can vary by technique and is dependent on the degree of knowledge required about the simulation. All optimization techniques posses their own internal parameters that must be tuned to achieve good performance. The time required to tweak these parameters is a part of the set-up cost. If the set-up for a particular technique requires a lot of prior knowledge about the simulation, one may be forced to delay its use until the later stages of the coercion process where the level of insight about the simulation is large. Alternatively, if the set-up time requires little or no insight, the optimization technique can be used in the early part of the coercion process.

The third criterion by which we compare optimization techniques is the degree to which the technique ensures a global minimum is discovered. Although local minima can be found efficiently, they provide little insight about the quality of results elsewhere in the

search space because a local minimum is merely the smallest value within a subsection of the search space. Figure 1 demonstrates the search space of a manufacturing process. The dark areas on the graph are the minima of the search space while the light areas are the maxima. For example, the medium-color spot centered about (4.5, 10.5) is a local minimum of the search space and the global minimum of the search space, the true minimum, is the dark area on the middle lower-half.

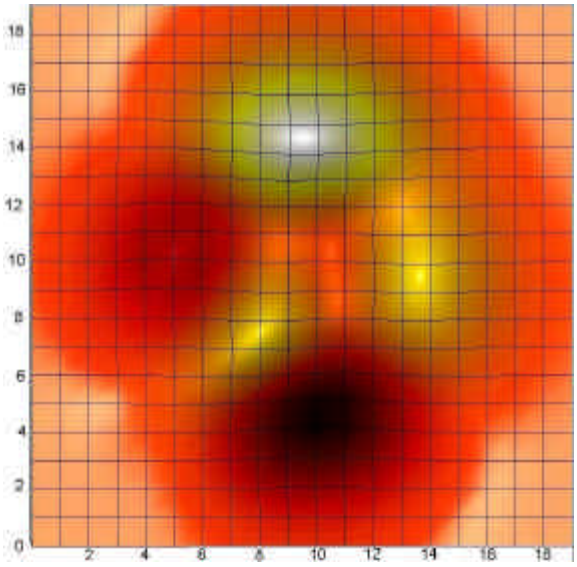

Figure 1: The search space of a manufacturing process.

Techniques that broadly explore the search space of potential decision variable values may fail to thoroughly explore pockets of the search space where promising results can be encountered, thus delaying the eventual convergence of the optimizer on the best results. This situation may be avoided by user insight that provides justification to focus exploration in a local region. However, this focused exploration may cause potentially better solutions that exist elsewhere not to be found. Because finding the global minimum depends on balancing exploration of what is unknown with exploitation of available insight, it is important for the user to determine where the system is well behaved and where it is unpredictable. Well-behaved systems are those that are non-chaotic and locally linear. A small change in input values will result in small changes in output. Many real-world problems are not adequately represented by well-behaved systems and require optimization techniques that explore the search space.

As decision variables are inserted or removed from the search space, the simulation's behavior can change dramatically, thus further complicating the user's awareness of how to strike a balance between exploration and guided local optimization. Accomplishing good results with coercion requires careful selection of an optimization technique that suits the user's insight regarding when, where, and how to search for a global minimum.

3.2 Heuristic Searches

Heuristic-based methods strike a balance between exploration and exploitation. This balance permits the identification of local minima, but encourages the discovery of a globally optimal solution [7]. However, it is extremely difficult to fine tune these methods to gain vital performance improvements [12]. Employing a heuristic search in the coercion process can prove very fruitful in terms of obtaining insight about the simulation because of its exploratory capabilities. Because of its exploitation capabilities, heuristic methods may also be able to obtain the best value for decision variables. Heuristic techniques are good candidate solutions when the search space is large and nonlinear because of their exploration capabilities. This characteristic makes it useful to use this class of techniques at the beginning of the coercion process, when insight is low. We review two popular heuristic methods below.

3.2.1 Simulated Annealing

Simulated annealing provides the user with an opportunity to combine exploitation and exploration. Exploitation comes from using gradient search, a simple algorithm that examines the nearby search space and moves towards the local minimum. Exploration comes from a stochastic element of the algorithm that causes deviation from the local minimum to other regions where improved solutions are possible. The stochastic nature of simulated annealing makes it well suited to find the minimum in systems that at not well behaved. The amount of randomness is controlled by two parameters to simulated annealing, the initial temperature and cooling rate. The initial temperature determines the level of randomness in the algorithm while the cooling rate determines how quickly the level of randomness decreases as the number of iterations of the algorithm increase.

Because of its exploration capability, simulated annealing is a good optimization technique to use where there are a large number of feasible solutions [11]. By monitoring the multiple regions explored by simulated annealing, a user can develop insight about the importance of specific decision variables, their regions of greatest influence on simulation behavior,

and their interactions with other variables. This insight contributes to the convergence of the coercion process. If the algorithm is left to iterate indefinitely, the temperature slowly decreases, causing the amount of exploration to decrease and resulting in discovery of the global minimum [9]. In the absence of user insight about the important local regions of the search space, it can be impossible to know with certainty that the global minimum has been found. Extended use of simulated annealing may not be an effective way to reduce the user's time in the coercion process if the user wants to find the value of the global minimum.

The initial set-up time of simulated annealing is low because the user can easily use educated guesses to assign values to the decision variables. However, the parameters to simulated annealing will most likely have to be fine tuned to discover optimum decision variable values and to explore the search space. Fine tuning the algorithms can take a significant amount of time and selection of the optimization technique's parameters can become an optimization problem in itself. For example, if simulated annealing is set-up with a low initial temperature and a high cooling rate, the algorithm has a low level of randomness that diminishes rapidly and hence the amount of exploration is low. However, if the initial temperature is high and the cooling rate is low, the level of randomness at the start is high and too much exploration is encouraged, possibly causing the algorithm to skip past areas where the global minimum resides. The user must find the right combination of values for the parameters of the algorithm for each simulation.

We illustrate the properties of simulated annealing in the coercion process by applying its use to an example. We present a potential run of simulated annealing with a high temperature and high cooling rate and the goal of finding the minima. According to the user's initial insight, the simulated annealing algorithm starts in the middle of the search space at (9, 9). Through gradient descent, it finds a minima at (10, 11). Because of the high level of randomness in the algorithm, the next iteration abandons the local minima and explores new regions by displacing the search to (7, 13). Gradient descent finds another local minima at (5, 10) and because the high cooling rate has caused the temperature to drop to a new level, the algorithm remains at the local minimum represented by the dark red area on the middle left part of Figure 1.

We now demonstrate simulated annealing on the same example, this time with a lower temperature and cooling rate and thus less randomness. The algorithm starts at the same place in the search space, at (9, 9). It descends to (10, 11) and because of its stochastic nature, gets displaced to (13, 8). Note, the lower temperature results in a much smaller displacement than the one that occurred with the previous example. The algorithm continues to converge and heads towards (12, 6) and then eventually to the global minimum displayed on the lower half of Figure 1.

Simulated annealing is traditionally used to find the global optimum [9] and to determine the best decision variable values. When the user has selected the right decision variables to optimize and constructed an objective function that correctly evaluates R', there is increased assurance that a global minimum will be discovered. If the user cannot specify the decision variables and objective function exactly, the algorithm provides an opportunity to explore the search space and gain insight. With a high temperature and low cooling rate, exploration is encouraged and the practitioner gains insight about the characteristics of the search space. Furthermore, exploitation within simulated annealing is good at finding local minima and educating the practitioner about the location of local minima within the search space.

3.2.2 Genetic Algorithms

Genetic algorithms is a heuristic search method derived from natural selection and evolution. At the start of a genetic algorithm optimization, a set of decision variable solutions are encoded as members of a population. There are multiple ways to encode elements of solutions including binary, value, and tree encodings. Crossover and mutation, operators based on reproduction, are used to create the next generation of the population. Crossover combines elements of solutions in the current generation to create a member of the next generation. Mutation systematically changes elements of a solution from the current generation in order to create a member of the next generation. Crossover and mutation accomplish exploration of the search space by creating diversity in the members of the next generation. This exploration is analogous to the stochastic nature of simulated annealing, but it differs in the opportunity to encode user insight in the crossover mechanism.

Traditional uses of genetic algorithms leverage the fact that genetic algorithms explore multiple areas of the search space to find a global minimum. Through the use of the crossover operator, genetic algorithms are particularly strong at combining the best features from different solutions to find one global solution. Through observation of these crossover combinations, the user gains insight about how parts of the simulation interact.

Genetic algorithms are also well suited for searching complex, highly non-linear spaces because they avoid becoming trapped in a local minimum [7]. Genetic algorithms explore multiple solutions simultaneously [6] and these sets of solutions make it possible for a user to gain multiple types of insight from one iteration of the algorithm. For example, one solution might provide the user with the insight that there is a correlation between a decision variable and the desired simulation behavior. Another solution may indicate to the user that a decision variable should no longer be considered because it has no impact on simulation behavior.

Genetic algorithms are difficult to set up because the user must choose how to represent and encode elements of a solution in the initial population. Additionally, they are difficult to fine tune because they may require modifying the set of decision variables, finding alternate solution encodings, and changing crossover and mutation implementations.

3.3 Gradient-Based Search

Gradient-based search methods are a category of optimization techniques that use the gradient of the objective function to find an optimal solution. Each iteration of the optimization algorithm adjusts the values of the decision variables so that the simulation behavior produces a lower objective function value. Each decision variable is changed by an amount proportionate to the reduction in objective function value. Gradient-based searches are prone to converging on local minima because they rely solely on the local values of the objective function in their search. They are best used on well-behaved systems where there is one clear optimum. Gradient-based methods will work well in high-dimensional spaces provided these spaces don't have local minima. Frequently, additional dimensions make it harder to guarantee that there are not local minima that could trap the search routine. As a result, as the dimensions (parameters) of the search space increases, the complexity of the optimization technique increases.

We illustrate the traditional use of gradient-based search on the search space illustrated by Figure 1. As in the previous example, the algorithm arbitrarily starts at the point (9, 9). By following the local gradient, the algorithm converges to the point (11, 10) and ends the search because the search process is trapped in a local minimum by the three peaks that surround it. The gradient-based search has converged rather quickly but it has not found the global minimum.

The benefits of the traditional use of gradient-based search techniques are that computation and set-up time are relatively low. However, the drawback is that global minima are likely to remain unfound. We advocate using gradient-based search techniques at the latter stages of the coercion process, after other search techniques have explored the search space and identified regions where the best of the local minima is likely to be a global minimum. For example, suppose simulated annealing is applied to a search space and through the algorithm's exploration a region near the global minimum at the bottom of Figure 1 is discovered. Once localized in that region, the user can employ a gradient-based search technique to quickly converge on the local minima, which in this scenario is the global minimum as well. This example efficiently combines the exploratory powers of heuristic-based search techniques with the computational efficiency of gradient-based search techniques.

Gradient-based search can be used at the start of the coercion process to gain insight about the local minima in the search space. The values that it finds may not be the best ones to transform a simulation but running multiple gradient-based searches with different starting points may be an effective way to determine the amount and placement of local minima.

3.4 Response Surface Methodology

Response surface methodology (RSM) is a statistical method for fitting a series of regression models to the output of a simulation model [7]. The goal of RSM is to construct a functional relationship between the decision variables and the output to demonstrate how changes in the value of decision variables affect the output. RSM is useful at finding the right combination of decision variables that will satisfy some specification. Relationships constructed from RSM are often called meta-models [12].

RSM usually consists of a screening phase that eliminates unimportant variables in the simulation [14]. After the screening phase, linear models are used to build a surface and find the region of optimality. Then, second or higher order models are run to find the optimal values for decision variables. Factors that cause RSM to form misleading relationships include identifying an incomplete set of decision variables and failing to identify the appropriate constraints on those decision variables.

RSM can be an extremely effective tool in the coercion process because it can be used to gain insight about the decision variables that affect specific behaviors of the

simulation. A user can also gain insight about the discontinuities in the simulation by examining the behavior of the simulation over a range of values for certain decision variables. This insight can be applied towards the set-up of another optimization technique. Insight gained about decision variables can also be used to discover correlations between code segments and simulation behavior. RSM may be difficult to work with because identifying the right set of decision variables and the right constraints requires a fairly high level of insight.

4. Conclusion

In this paper, we have motivated the role of insight as it relates to simulation coercion. By measuring the impact insight has on the effort required to coerce a simulation, we provide a formal means to quantify progress towards satisfying transformation requirements. The formalism takes the form of a coercion invariant, $D(S_i, I_i) < D(S_{i-1}, I_{i-1})$, which provides two benefits. First, the invariant provides a guarantee that coercion will converge on a solution. By providing such a guarantee, coercion is a viable alternative to other transformation technologies. Secondly, the quantification of insight provides a means to evaluate the suitability of computational tools to the coercion process.

We evaluate optimization techniques, a class of computational tools frequently used in the simulation community. We classify these techniques with theoretical motivations in mind and identify ways they assist in gathering insight during the coercion process. The utility placed on insight gained may be an unconventional measurement compared to the typical evaluation of optimization techniques, and the review of optimization techniques demonstrates many ways that insight dramatically influences the transformation process.

Our results serve as a guide to combine optimization and manual code modification in the coercion process. Such a guide is needed because simulations typically have too many undetermined variables to be tractably solved with monolithic optimization techniques and manual code modification presents the user with too many complex interactions between decision variables and simulation behavior. Neither technique can be used exclusively. Through the use of both techniques in the coercion process, user insight can be generated and validated in order to maximize the benefits of each.

5. Acknowledgements

We wish to acknowledge support from the Defense Modeling and Simulation Office, particularly from Sue Numerich and Phil Zimmerman. Additional support was provided by the National Science Foundation (ITR 0426971).

6. References

[1] P.F. Reynolds: "Using Space-Time Constraints to Guide Model Interoperability" Proceedings of the 2002 Spring Simulation Interoperability Workshop, 2002.

[2] S. Waziruddin, D. C. Brogan, P. F. Reynolds: "The Process for Coercing Simulations" Proceedings of the 2003 Fall Simulation Interoperability Workshop, 2003.

[3] D. T. Drewry, P. F. Reynolds and W. R. Emanuel: "An Optimization-Based Multi-Resolution Simulation Methodology" Proceedings of the 2002 Winter Simulation Conference, 2002.

[4] J. C. Carnahan, P. F. Reynolds and D. C. Brogan: "An Experiment in Simulation Coercion" Proceedings of the 2003 Interservice/Industry Training, Simulation and Education Conference, 2003.

[5] M. C. Fu, S. Andradottir, J. S. Carson, F. Glover, C. R. Harrel, Y. Ho, J. P. Kelly, S. M. Robinson: "Integrating Optimization and Simulation: Research and Practice" Proceedings of the 2000 Winter Simulation Conference, 2000.

[6] S. Olafsson and J. Kim: "Simulation Optimization" Proceedings of the 2002 Winter Simulation Conference, 2002.

[7] Y. Carson and A. Maria: "Simulation Optimization: Methods and Applications" Proceedings of the 1997 Winter Simulation Conference, 1997.

[8] S. Andradottir: "A Review of Simulation Optimization Techniques" Proceedings of the 1998 Winter Simulation Conference, 1998.

[9] F. Azadivar: "Simulation Optimization Methodologies" Proceedings of the 1999 Winter Simulation Conference, 1999.

[10] F. Glover, J. P. Kelly and M. Laguna: "New Advances for Wedding Optimization and Simulation" Proceedings of the 1999 Winter Simulation Conference, 1999.

[11] J. R. Swisher, P. D. Hyden, S. H. Jacobson, L. W. Schruben: "A Survey of Simulation Optimization Techniques and Procedures" Proceedings of the 2000 Winter Simulation Conference, 2000.

[12] M. C. Fu: "Simulation Optimization" Proceedings of the 2001 Winter Simulation Conference, 2001.
[13] K. C. Louden: "Formal Semantics of Programming Languages" Programming Languages: Principles and Practice, PWS Publishing Company, Boston 1993.
[14] R. H. Myers and D. C. Montgomery: Response Surface Methodology: Process and Product Optimization Using Designed Experiments, Wiley-Interscience, 2002.

Author Biographies

SARAH WAZIRUDDIN is a graduate student in the Computer Science Department and a member of MaSTRI at the University of Virginia. She is an expert in simulation coercion and user interfaces.

DAVID BROGAN earned his PhD from Georgia Tech and is currently an Assistant Professor of Computer Science and a member of MaSTRI at the University of Virginia. For more than a decade, he has studied simulation, control, and computer graphics for the purpose of creating immersive environments, training simulators, and engineering tools. His research interests extend to artificial intelligence, optimization, and physical simulation.

PAUL F. REYNOLDS, Jr. is a Professor of Computer Science and a member of MaSTRI at the University of Virginia. He has conducted research in modeling and simulation for over 25 years, and has published on a variety of M&S topics including parallel and distributed simulation, multi-resolution models and coercible simulations. He has advised numerous industrial and government agencies on matters relating to modeling and simulation. He is a plank holder in the DoD High Level Architecture.