Proceedings of the 2007 Winter Simulation Conference
S. G. Henderson, B. Biller, M.-H. Hsieh, J. Shortle, J. D. Tew, and R. R. Barton, eds.

AGILE OPTIMIZATION FOR COERCION

Lingjia Tang,
Paul F. Reynolds, Jr.

Department of Computer Science
University of Virginia
Charlottesville, VA 22904, U.S.A

.

ABSTRACT

C*oercion* combines flexible points, semi-automated optimization and expert guided manual code modification for adapting simulations to meet new requirements. Coercion can improve simulation adaptation efficiency by offloading large portions of work to automated search. This paper identifies requirements and related challenges in coercion, presents methods for gaining insight, and describes how to use these insights to make agile strategy decisions during a coercion. We call our optimization method *agile optimization*, because it allows users to preempt optimization and flexibly interleave alternative optimization methods and manual code modification, as needed. Agile optimization exploits the combined strengths of human insight and process automation to improve efficiency. We describe a prototype system and a case study that together demonstrate the benefits that can accrue from agile optimization.

1 INTRODUCTION

COERCE is a semi-automated technology for adapting simulations to meet new requirements. A necessary part of constructing a model is selection of abstractions, in order to reduce complexity, improve performance, and provide estimations for unknowns. When developing *coercible* simulations a subject matter expert identifies a set of *abstraction opportunities and alternatives* for each model abstraction. A *flexible point* of a simulation reflects one model abstraction opportunity and the corresponding *bindings* for the flexible points reflect abstraction alternatives. Flexible points can include both parameters and model abstraction opportunities which reflect design decisions, implementation considerations, and model uncertainties. Flexible points can be annotated and constrained using formal methods (Carnahan 2006). The coercion process is as follows: when a new set of requirements for a simulation arises, the user needs to determine whether to adapt the simulation by numerical optimization or manual code modification. For optimization, the user needs to define the objective function for optimization to represent the difference between the simulation's current behavior and required new behavior. Also, the user should identify related flexible points, which would become the variables over which an optimization is conducted. Bindings for each flexible point represent the valid search range during optimization. Optimization automatically searches for appropriate bindings of flexible points that minimize the objective function, thus minimizing the distance between the simulation's current behavior and required behavior. At those times that optimization would fail to find a satisfactory solution, manual code modification can be employed. A user monitoring the coercion must acquire sufficient insight to determine a next course of action. The following notation depicts the iterative nature of the COERCE process:

$$S_0 \rightarrow_p S_1 \rightarrow_p \ldots \rightarrow_p S_{n-1} \rightarrow_p S_n$$
$$p = o \mid m$$

where S_0 is the original simulation, S_n is a simulation instance meeting the new requirements, and $\rightarrow_p$ is either an optimization o, or a manual modification m to the source code (Waziruddin, Brogan and Reynolds 2004).

Optimization over simulation inputs ("simulation optimization") to meet certain output objectives has been employed for years. Coercion differs from simulation optimization (SO) in part because it uses optimization to transform simulations themselves, and because it employs flexible points to guide and constrain optimization. Flexible points, which are the source of variability in coercion, tend to be much richer than traditional parameter sets. SO is typically applied to parameter sets only. Flexible points also include structural alternatives. Furthermore, flexible point notation supports expression of dependencies and interactions between flexible points

(incrementally) provided by subject matter experts and coercion users, thus capturing constraints which can greatly reduce optimization search space. The following requirements exist for coercion but have not existed for traditional simulation optimization:

- The optimization portion of coercion must provide insight into choices for possible simulation implementation adaptations.
- Because coercion combines optimization and manual code modification to adapt simulations, it must be an iterative process. Traditionally SO is applied monolithically.
- Because user insight is essential in coercion, internal processes must be exposed, in order to enhance insight and to provide guidance for later steps in the coercion process. In traditional SO simulations are treated as black boxes. Coercion must treat them as white boxes.
- Agility in the coercion process is essential. Having optimizations run to completion and/or repeatedly executing a simulation to completion can consume unacceptably large amounts of time. During any coercion step, $S_{i-1} \rightarrow_p S_i$, a user must be able to easily switch to another optimization method or manual modification.

In partial fulfillment of these requirements, we propose *agile optimization*, an approach that supports preemption and replacement of a coercion step when an optimization is not proceeding as expected. With agile optimization, a user selects an optimization method and initiates a search. During the search, the user monitors the process, gathers insight through visualization and analysis support mechanisms, and adjusts strategies and tactics based on the insights. A user can preempt the process at any time to implement a strategic or tactical change. Possible changes include adding or modifying flexible points, adding insights about flexible points, switching to another optimization method, tuning the current optimization method, modifying the desired outcome, or returning to manual modification. The framework supporting agile optimization provides visualization, statistical analysis tools and controls for users to gain insight and guide the coercion process. Also, it facilitates preemption and switching of an ongoing optimization, and it provides support for rapid integration and exploitation of a user's insights. Thus, a user can expect to be able to chain optimization techniques sequentially for exploration, and to interleave optimizations and manual code modification flexibly. This approach exploits a human user's superiority in pattern recognition, learning and decision making while also exploiting the advantages of automated search.

Agile optimization is fundamental to achieving the requirements for coercion listed above:

- It is designed to exploit human insight and involvement in the coercion process, and it includes provisions for guiding coercion using that insight.
- It combines the respective strengths of humans and computers to improve quality and efficiency in the coercion process.
- It supports preemption of and chaining of multiple optimization techniques in order to maximize efficiency and acquisition of insight.
- It utilizes interaction indications and constraints related to flexible points to limit search space.

Finally, we have constructed a prototype system that includes basic visualization tools and a small library of optimization techniques to provide users some flexibility in conducting simulation coercion. While the general system is still under construction and considerable research and development remains, we present a case study using the prototype system that demonstrates the evolving system's efficacy.

2 PREVIOUS WORK

Coercion has been applied successfully to adapt simulations in different domains, as reported in Drewry, Reynolds, and Emanuel (2002), Carnahan, Reynolds, and Brogan (2003), and Carnahan and Reynolds(2006). For example, Drewry, Reynolds, and Emanuel (2002) describes the coercion of a low-resolution simulation of carbon dioxide uptake in forests to yield behavior closely matching the behavior of a trusted high-resolution model. In these studies, automatic optimization was used to find appropriate bindings for flexible points that yield a desired simulation behavior. However, there has been no research on how to flexibly interweave automatic optimization and manual code modification. Also, no human interaction was introduced into the optimization process, and users were provided only an ad hoc method for choosing an optimization technique. Key contributions of this paper include its approach to increasing user insight, leveraging users' insights and strategic thinking in the optimization process and flexibly chaining various techniques in the coercion process.

In the optimization community, research has occurred on interactive optimization where a human is brought into the loop and is assigned different tasks such as modifying the solutions or eliminating unsatisfactory or unrealistic solutions. Interesting work includes Human-guided tabu search (Klau et al. 2002) and the HuGs platform (Klau, et al. 2002b). Several optimization techniques and solution visualizations are provided in HuGs. There, users can preempt a current optimization, modify a solution, constrain a search, or switch to another technique. With HuGs, experiments showed that human-guided optimization outperformed an equivalent amount of

unguided optimization. However, the HuGs work is limited to a special class of combinatorial problems and a limited set of optimization techniques (tabu search, exhaustive search and greedy search).

Early research has been conducted on combining various optimization techniques. Hogg and Williams(2003) have proposed a cooperative search paradigm where two or more search algorithms are running in parallel and exchanging and reusing information generated by the other. Faulkner and Cowart (2006) present an optimization method that interleaves thirty different algorithms to solve nonlinear optimization problems.

Human interaction with optimization has been explored in the simulation community recently. Persson, Grimm, and Ng (2006) propose a white-box optimization approach. They advocate using visualization and statistics about the optimization process to help users tune the technique and gain insight. This work focuses on observing an optimization and tuning the current optimization technique. Our work focuses on gaining and leveraging insights into both optimization and a simulation. Also, we advocate agility in the method heretofore unexplored.

3 APPROACH

Agile optimization proceeds as depicted in Figure 1. Given a set of flexible point definitions and a new set of requirements, the user must determine whether the best next step involves optimization or manual code modification. For optimization, the user must specify an objective function and constraints that represent the new desired behavior, identify an appropriate set of flexible points as the optimization variables and select an optimization technique to start. During a search, the user observes the process, and gathers insights from visualization, analysis, etc. During a search, or at the end of one, the user will evolve his/her strategy for proceeding.. A strategic decision may include preemption and chaining various optimization techniques flexibly. Others could be:

- tune the current search's parameters or configuration and restart
- modify the flexible points, or add insights about them
- constrain the search space according to insights or modify the objective function and restart.
- switch to manual code modification
- modify the solution (change the bindings of flexible points in the solution) and continue

The process continues until the user is satisfied with the results. The key contributions of our approach relate to our goal of providing insight in a process based manner. This process, as noted, includes the use of flexible points, optimization algorithm agility, and search.

A framework for agile optimization should include several essential components:

- support for cumulatively gaining insight into simulation behavior and flexible points during the agile optimization process. Examples include visualization and statistical analysis components.
- support for carrying out strategic changes and integrating insights, including:
  - a large set of optimization techniques that are preemptible and tunable, and methods for communicating between techniques in order to control the overhead costs associated with chaining techniques.
  - a user-friendly interface for users to observe, control and enforce new strategies interactively.

Figure 2 shows the architecture for the framework. In the following sections, we discuss details of the proposed architecture.

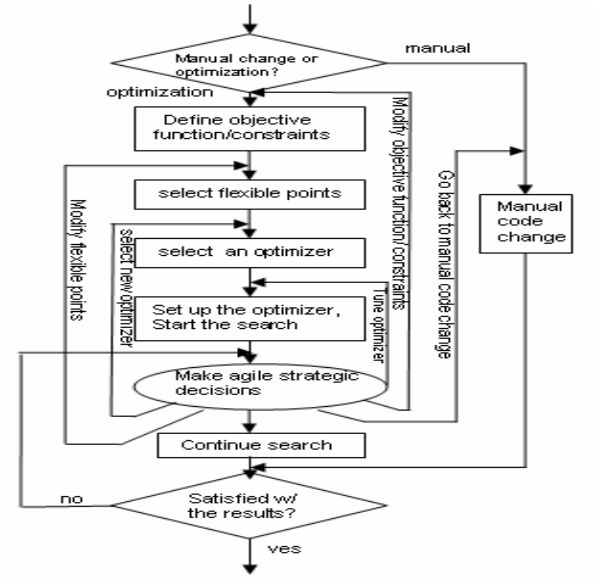


Figure 1: Process of Agile Optimization

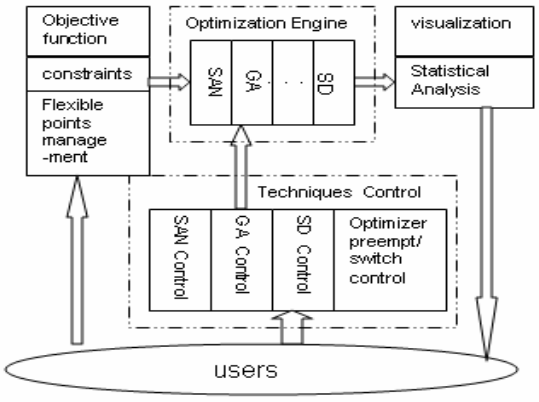


Figure 2: Architecture for Agile Optimization. SAN: Simulated Annealing; GA: Genetic Algorithm; SD: steepest descent. Users can enforce strategy changes

through technique control and objective function/flexible point management. The status of an optimization is presented to the user through visualization and statistical analysis components.

3.1 Insights

Insights are essential in our approach. Given sufficient assistance, as would occur in an agile optimization framework, a user can gain useful insights that in turn reveal promising directions and effective strategies for coercion. Such insights are generally not available as a user begins to adapt a simulation. With agile optimization the goal is to make them apparent as early as possible, so that the adaptation can proceed efficiently. Because flexible points have semantics representing abstraction alternatives in the simulation (Carnahan, 2006) they create more opportunities to gain valuable insights about themselves and their effects on simulation behavior. Moreover, newly-gained insights about flexible points and simulation behavior can be annotated and used for coercion in the future (Carnahan, 2006).

Insights can be used to guide a successful coercion. They can help to constrain the search space. They can also indicate a need for strategy change as well as suggesting appropriate changes. Insights are central to problem redefinition, process performance improvement and search technique selection. (Tang and Reynolds 2007) present a summary of several common optimization techniques and describe possible insights each technique can provide and important insights each technique needs for appropriate setup. Their summary can serve as a user guide for exploiting insights to chain optimization techniques and to guide the process. Examples of insights occurring during agile optimization include:

- Effects of flexible points on variables of interest and/or the objective function
- Importance and/or sensitivity of flexible points
- Interactions between flexible points which can be used to reduce the search space.
- Promising regions where search efforts should be intensified .
- Problem structure/ characteristics of the landscape
- Current search success (stuck in local minimum, searching a poor area, not converging) and possible changes.
- Insights about the objective function definition, new constraints, etc.

There are several approaches for gaining insights. Most occur as a result of observing near successes and analyzing the causes of failures during a search. We believe more knowledge leads to more insight, and speedy acquisition of insight is important. Agile optimization is designed to provide speedy acquisition of insight.

Insight can be gained from visualization of intermediate optimization results, partial executions of the subject simulation as well as the simulation's current behavior compared to the target behavior. Also, statistical analysis can be used to help gain insights into both the interactions between flexible points and effects/importance/sensitivity of flexible points. It can be used together with visualization into the specific technique's characteristics (e.g diversity of population in genetic algorithm, system temperature in simulated annealing) to give insights about how to tune the optimization.

The importance of insight cannot be overstated. In the next section we demonstrate its utility in shaping an agile approach to optimization

3.2 Agile Strategies

Possible changes in strategy are listed at the beginning of this section. Users steer the coercion process based on insights. Example scenarios include:

- Optimization may enter a path that eventually generates no satisfactory results. A user may realize that and preempt the process to adjust the strategy.
- A user may realize no optimization method is going to succeed. In this case, the best choice is likely preemption, to begin code modification.
- The user-defined objective function might not capture complex requirements sufficiently. User interaction is needed to steer the optimization and coercion process agilely.
- When a user has insights about how to improve a current strategy, s/he should leverage the insights as soon as possible.
- A user may realize that a selected optimization method is inappropriate (e.g. too powerful and thus too slow) for current conditions.

To support agility, the agile optimization system must provide tools for speedy acquisition of insight and optimization techniques that are easy to manipulate and switch. We discuss the techniques more in section 3.3. Finally, heuristics can be employed to predict the need for strategy change and provide suggestions.

A human/computer division of workload exploits automated computation power especially on constrained and focused search spaces. It also leverages a user's domain knowledge and exploits the human advantage in pattern recognition, learning, strategic thinking and decision making (Klau, et al. 2002). Literature on interactive optimization shows that humans are very good at deciding when to preempt optimization and identify promising areas of the search space (Scott, Lesh, and Klau 2002), which further supports our approach. Although some optimization techniques have adaptive schemes to

adjust to different problems, we argue that human learning skills are still required for best performance. Finally, including a user in the process assists the user in gaining confidence in the correctness of the coerced results.

3.3 Optimization Strategies

We have focused on chaining optimization techniques for coercion. To take full advantage of flexible points, multiple optimization techniques should be available. Various optimization techniques have selected strengths and weaknesses. Some require little insight or setup cost, while others can exploit detailed information about the simulation to improve quality of results. Different techniques produce different search trajectories thus yielding different insights. Chaining can exploit strengths of different techniques, solve larger classes of problems, leverage insight and help users gain different insights.

To support chaining, optimization techniques must be modified so that their controlling parameters can be tuned on-the-fly and they can be preempted and resumed with different configurations specified by users. Communication between techniques should be facilitated to support switching. Search space information should be retained so that a switched-to technique can start where the previous technique ended. A list of best-ranking solutions should be kept for a user to consider.

Optimization techniques should use user insights as guidance for search directions, to concentrate on promising regions, reduce search space, avoid invalid regions, etc. Techniques should be modified and provide visibility and control for the following kinds of insights:

Effects/importance/sensitivity of flexible points

New constraints – Users should be able to add new constraints on the fly.

Promising regions/direction – When a promising region is discovered, a user should be able to guide search accordingly (e.g. bounding flexible point ranges). Upon discovery of a complimentary relationship between flexible points, the optimization technique can generate trial solutions accordingly.

Interactions between flexible points that bound the search space – Incompatible or canceling effects between flexible points can be used to reduce the search space. Flexible point combinations can be disallowed or their probability reduced.

We have built a prototype based on the approach presented here and conducted preliminary case studies. We describe the prototype and one such study next.

4 CASE STUDY

4.1 Prototype System

Our system uses tools and a formal language(Flex ML) developed by Carnahan, Reynolds and Brogan (2005) for documenting flexible points and insights about them. For each iteration of optimization, our system

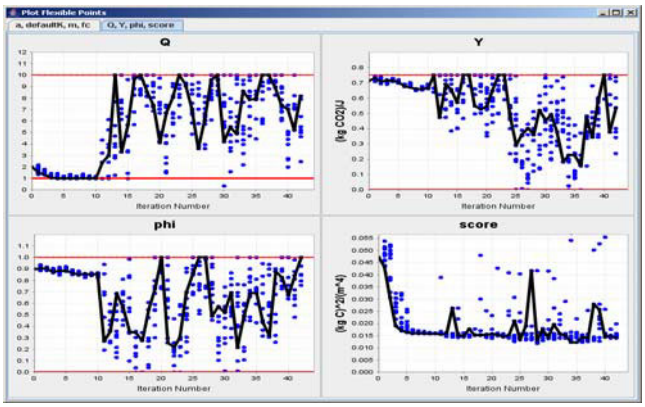


Figure 3: Sample Visualizations. Graphs Q, Y, and phi show the coverage of the search space for the three named flexible points over 45 iterations. Blue dots show the range of bindings in each iteration that the optimizer evaluates. The black line is the binding of the selected solution, used for subsequent iterations. Graph "score" shows related evaluation of the objective function (lower is better). The blue dots are evaluations of all trail solutions that the optimizer generates in each iteration. The black curve shows the evaluation of the selected solution. Simulated annealing was the selected optimization method, which is why a best solution is not always selected.

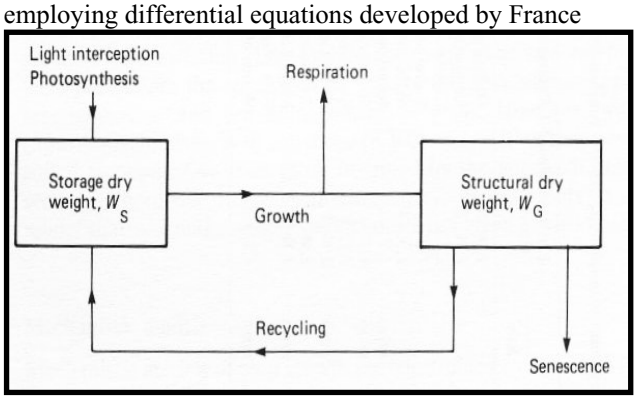
executes a new simulation instance with selected bindings for relevant flexible points, and evaluates the objective function. Available optimization methods include simulated annealing, a genetic algorithm and steepest descent. User insight about flexible points is exploited. A user can suggest promising search directions and avoid poor/invalid combinations of flexible point bindings in a search. A user can incrementally add newly-gained insights during the process to be used immediately.

The system provides dynamic visualization of the effects of an optimization, which helps a user gain insight about, for example, how flexible points' selected bindings affect outcomes as the optimization proceeds, as well as how different bindings for flexible points affect the objective function. An example of a running profile is shown in Figure 3. A user can preempt, restart with different configurations or switch to other techniques. A user can also manage flexible points: add or delete flexible points, change their boundaries and add newly-discovered interactions between flexible points. Finally, a user can modify the solution then continue the search.

4.2 Coercing Grass Growth Simulation

4.2.1 Grass Growth Simulation

The Grass Growth simulation is based on a model employing differential equations developed by France



$$\frac{dW_S}{dt} = \phi\theta P - \frac{\mu W_G}{Y} + \gamma W_g \qquad \frac{dW_G}{dt} = \mu_m \frac{W_G W_S}{W_G + W_S} - \gamma W_G - \beta W_G$$

Figure 4 Grass Growth Model and Two main differential equations in the model

(1984) to study crop growth. The model includes five equations and thirteen parameters which are all identified as flexible points in the model by a subject matter expert. The expert didn't identify structural flexible points or interactions between the flexible points in this model. Still it is possible for a coercion user to identify those during the coercion process.

The environmental inputs of the simulation are day light information, temperature and length of days. The output of the simulation is the dry weight of the crop (sum of storage dry weight W_s and structural dry weight W_g).

4.2.2 Case Study: Coercing Grass Growth Simulation

The goal of our coercion exercise is to coerce the Grass Growth simulation to match the real-world grass data.

We used the Hurley grass data collected in Finland, 1983(IGER 1983). For every week during this year, sun and sky radiation, sunshine hours, temperatures and grass weight were measured. As input to our Grass Growth simulation we use the real-world inputs supplied by the dataset and we output grass weight. The initial comparison of our simulation's output and the real world grass weight data is shown in Figure 5. As Figure 5 shows our simulation's output (dashed red) does not match the real world data (solid black). This motivates the coercion: using real world input data we will semi-automatically coerce the simulation to produce similar grass growth data similar to real world grass data.

Coercion Process:

1) The user provided a visualization comparing real-world data and output data, in Fig 5, which also shows the simulation's original behavior. To match these two curves, the user defined the objective function to be minimization of the variance in differences between real-world data and simulation output data. The differences are sampled on a daily basis for a year. Minimizing variance

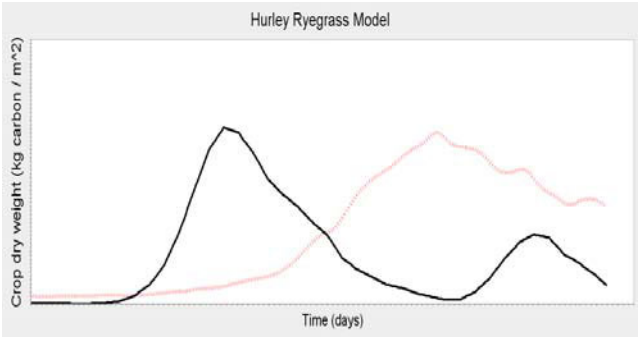


Figure 5: Target grass growth curve(solid black) vs. simulation's output(dashed red) before coercion started. X-axis is day number during a year. Measured daily data is interpolated to be smooth and comparable with continuous Grass Growth simulation

is a preferred approach for matching the output curve with the shape of the target curve.

2) The user selected seven relevant flexible points from the flexible description file. All selected flexible points have uncertain boundaries and default values. The selected flexible points are as follows:

a :	Structural specific leaf area
k :	Extinction coefficient of canopy
fc:	Fraction carbon content of dry matter
Q :	Q_10 factor for rate constants
Y :	conversion yield of growth
ϕ :	shoot growth allocation
m :	Leaf transmission coefficient

Objective function:

minimize $Var(realData_i - simulationData_i)$,

$i = 1, 2...365$ ..(i *is day number in a year, two data are compared on daily basis. Real-world data is interpolated to generated daily data, simulation outputs daily data*)

3) The user selected simulated annealing with its default parameter settings and decided to use default values of flexible points as the start point, and initiated the search.

4) The user observed the running profile visualization and realized that objective function evaluation is better when variables a , k are close to their upper bounds. When the search terminated, the user checked the visualization (Figure 6) and was not satisfied with the results.

Flexible Points	$\min_1$	$\max_1$	$start_1$	$result_1$		$\min_2$	$\max_2$	$start_2$	$result_2$		$\min_3$	$\max_3$	$start_3$	$result_3$
a	10	30	21	30.00		10	40	21	38.778		10	40	38.778	40
k	0	1	0.5	0.587		0.001	1.5	0.5	1.5		0.6	1.5	1.5	1.5
m	0.01	0.95	0.12	0.010		0.01	0.95	0.12	0.01		0.01	0.95	0.01	0.254
fc	0.01	1	0.4	0.809		0.01	1.5	0.4	0.622		0.01	1.2	0.622	0.693
Q	1	10	2	9.585		1	20	2	18.677		4	20	18.677	6.836
Y	0	0.75	0.715	0.324		0.001	0.75	0.715	0.254		0.001	0.6	0.254	0.290
ϕ	0	1	0.9	0.729		0.001	1.5	0.9	0.722		0.3	1.5	0.722	1.29
initial W_s			0.01			0.001	0.03	0.01	0.003		0.001	0.03	0.003	0.001
initial W_g			0.01			0.001	0.03	0.01	0.03		0.015	0.03	0.03	0.028
Objective Function				0.0118					0.00516					0.0023

Table 1. Bindings of flexible points in each optimization phase. Subscript number indicates the phase: 1 is simulated annealing used in step 3); 2 is simulated annealing used in step 5); 3 is the steepest descent used in step 6). min and max are lower and upper bounds for flexible points bindings. Start and result are the start binding and result binding of the flexible point for the search.

shows in step 5, users expanded boundaries and added flexible points.

shows the in step 6, users constrain the boundaries to focus the search and the new start point.

shows the result bindings from each optimization phase

the last row shows the evaluation of objective function of the best result from these three optimization phases.

5) The user returned to the simulation, and decided to extend the upper bounds of a , k (the flexible point description file indicated boundaries are uncertain). The user also noticed that in the optimization results, variables fc, Q , ϕ 's bindings are close to their boundaries, so a decision was made to extend their upper bounds as well. The user also re-examined the grass-growth model and decided to include two more flexible points into the optimization:

$initial W_s$: initial storage dry

$initial W_g$: initial structural dry weight

After these modifications, which are shown in table 1, the user initiated simulated annealing again.

6) When the search terminated, the user was still slightly not satisfied with the results (Figure 7). The user suspected that simulated annealing was cooling down too fast. And the user decided to switch to steepest descent to search for local minima around the result of the simulated annealing. If the local minimum is satisfactory, this strategy is more efficient than tuning simulated annealing. The user also noticed from the visualization that flexible points k , fc, Q , ϕ , Y , initial W_g remain within sub-ranges of their valid domain (an example is in Figure 8). The user decided to focus the search in these sub-ranges. The user used the best solution from simulated annealing as the start point for steepest descent, as shown in Table 1.

7) After steepest descent terminated, the user checked the solution and curves, and was satisfied (Figure 9). The flexible points setting and management in each

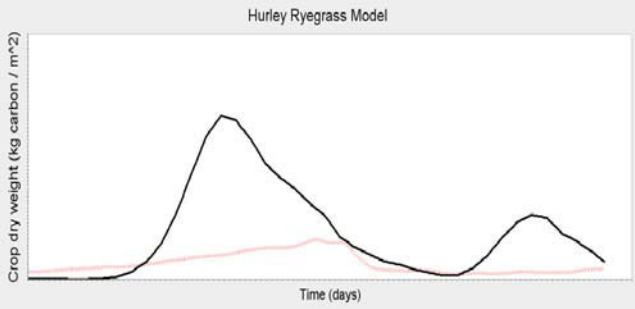


Figure 6: after first simulated annealing search in step 3

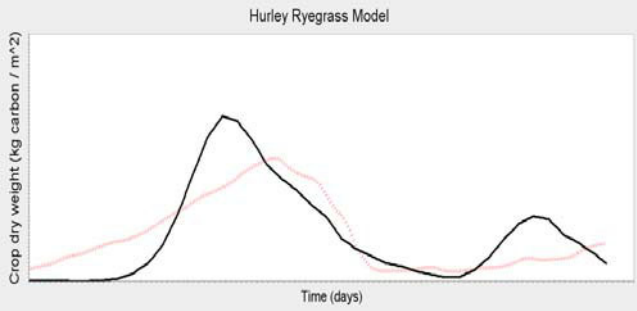


Figure 7: after second simulated annealing in step 5

optimization phase are shown in Table1. The result of each optimization phase is shown in figures 6, 7 and 9.

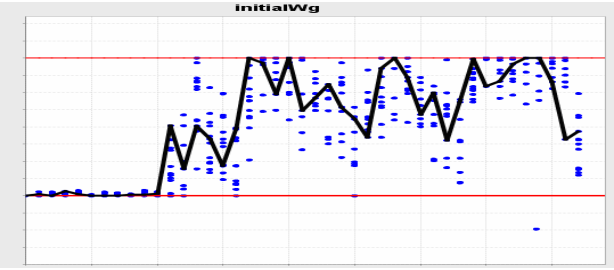


Figure 8: initial W_g. As the optimization proceeds, its values in the sample solutions are moving from around lower-bound to a sub-range closer to upper bound.

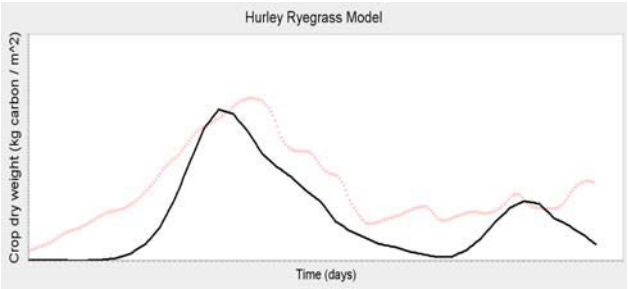


Figure 9: after steepest descent in step 6

4.2.3 Discussion

In this case study, the user made effective strategy decisions guiding the coercion process. The user identified two new relevant flexible points, and identified a promising direction for expanding the boundaries of several flexible points during the process and restarted the search. These decisions improved the results greatly (from Figure 6 to Figure 7 and to Figure 9) according to the stated objective. The user also switched to local search and focused the search in a promising sub-search space which also improved the results (from Figure 7 to Figure 9).

It should be noted that our focus is not on finding the 'only' or the 'optimal' strategic decisions. Our goal is to study the efficacy of strategic user guidance of a successful coercion. In some situations, users have more than one decision option. For example, in this exercise, steepest descent found a set of satisfactory results efficiently. Alternatively the user could have chosen to tune the simulated annealing or restart simulated annealing instead of switching to steepest descent. Also, the user could have made mistakes. For example, the user expanded the boundary of Q, but the binding of Q in the result solution is within its original bound. However overall, the user made effective strategy decisions when the optimization did not give satisfactory results.

The user did not preempt the optimization in the case study. The main reason is that the Grass Growth simulation is not time-consuming. Also, the optimization ran for a short period of time and the results continuously improved as expected during the process. However, for long running optimizations we provide dynamic visualizations and allow users to preempt the optimization process at any time if the optimization process ceases to be productive.

5 FUTURE WORK AND CONCLUSION

Agile optimization and coercion offer many advantages over more traditional simulation optimization methods, including:

- Exploitation of flexible point annotations as well as indications of flexible point interactions and constraints. This can reduce the search space considerably and provide increased insight as optimization occurs.
- Support for agility in switching between modification and optimization and among optimization methods. Support for agility can mean greater flexibility and increased efficiency.

Coercing simulations is a powerful adaptation technique, however it poses several challenges. We have presented an interactive approach where users employ insights gained during the optimization portion of the coercion process to make agile strategic decisions to guide the whole process. The contributions of this paper include:

- Identification of the main challenges in the coercion process
- Identification of the importance of users' insights and interaction in coercion. We provide a preliminary taxonomy of insights and we discuss tools to assist users in gaining insight.
- Definition of an interactive approach where users can preempt the coercion process and make agile strategic decisions. This approach reflects respective strengths of humans and computers.

- Focus on chaining optimization techniques using optimization algorithm-specific transition information we have described previously (Tang and Reynolds, 2007).

This case study has motivated several opportunities for future work. More sophisticated visualizations should be provided to help users gain more complex insights about interactions between flexible points. Also, statistical analysis is not conducted in this case study. It could be helpful for users to gather insights regarding the importance and sensitivity of flexible points and how the current automated search technique is progressing. Such insights are likely to provide guidance for preempting the coercion process. Designing better visualization methods and conducting experiments using statistical analysis to gain insights are interesting future research topics. We also plan to conduct more studies on techniques to improve the agility of strategic decisions in the process. For example, a dynamic sensitivity analysis on the effectiveness of the current optimization technique would identify good locations for preemption. Finally, we are interested in minimizing the overhead of chaining various optimization techniques in order to increase the efficiency of the coercion process.

REFERENCES

Carnahan, J. C. 2006. "Language Support for the Coercible Software Domain". A Dissertation Proposal, University of Virginia: School of Engineering and Applied Science, Charlottesville, VA, 2006.

Carnahan, J. C., and P. F. Reynolds. 2006, December. "Requirements for DDDAS flexible point support". In *Proceedings of the 2006 Winter Simulation Conference*, ed. L. F. Perrone, F. P. Wieland, J. Liu, B. G. Lawson, D. M. Nicol, , and R. M. Fujimoto. Institute of Electrical and Electronics Engineers, Inc.

Carnahan, J. C., P. F. Reynolds, Jr., and D. C. Brogan. 2003. "An Experiment in Simulation Coercion." In *Proceedings of the 2003 Interservice/Industry Training, Simulation, and Education Conference*, December 2003.

Carnahan, J. C., P. F. Reynolds, Jr., and D. C. Brogan. 2005. "Simulation-Specific Characteristics and Software Reuse." In Proceedings of the 2005 Winter Simulation Conference, pp. 2492-2499, December 2005.

Drewry, D. T., P. F. Reynolds, W. R. Emanuel. 2002. "An Optimization-Based Multi-Resolution Simulation Methodology". In *Proceedings of the 2002 Winter Simulation Conference*, December 2002.

Faulkner, E. and J. Cowart. 2006. "The Adaptive Optimization Engine", Presented at INFORMS Annual Meeting 2006.

France, J. 1984. "Mathematical Models in Agriculture: A Quanitative Approach to Problems in Agriculture and Related Sciences", Buttersworths Publishing, Boston, MA, 1984.

Hogg, T. and C. P. Williams. 2003. "Solving The Really Hard Problems With Cooperative Search", In *Proceedings of the AAAI93*, AAAI Press. Menlo Park, CA,1993.

IGER (Institute of Grassland and Environmental Research), 1983. "Hurley Grass Dataset". Personal communication <Scott.Laidlaw@afbini.gov.uk>

Klau, G. W., N. Lesh, J. Marks, and M. Mitzenmacher. 2002. "Human-Guided Tabu Search." In *Proceedings of the 18th National Conference on Artificial Intelligence(AAAI)*, pp. 41-47, 2002.

Klau, G. W., N. Lesh, J. Marks, and M. Mitzenmacher. 2002b. "The HuGS Platform: A Toolkit for Interactive Optimization" , Published in *Advanced Visual Interfaces*, May 2002. Trento, Italy.

Persson, A., H. Grimm, A. Ng. 2006. "On-line instrumentation for simulation-based optimization". In *Proceedings of the 2006 Winter Simulation Conference*, December 2006.

Scott, S. D., N. Lesh, G. W. Klau. 2002. "Investigating Human-Computer Optimization". In *proceedings of Conference on Human Factors in Computing Systems*, 2002.

Tang, L and P.F. Reynolds 2007. "User guide for chained use of optimization techniques to gain insight", *Technical Report: CS-2007-03*, University of Virginia, Computer Science Dept, Jan. 2007.

Waziruddin, S., D. C. Brogan, P. F. Reynolds. 2004. "Coercion through Optimization: A Classification of Optimization Techniques". In *Proceedings of the 2004 Fall Simulation Interoperability Workshop*, Orlando, FL, September 2004.

Wolpert, D.H., Macready, W.G. 1997. "No Free Lunch Theorems for Optimization", *IEEE Transactions on Evolutionary Computation*, VOL. 1, NO. 1, April 1997.

ACKNOWLEDGMENTS

We gratefully acknowledge support from the DDDAS program at the National Science Foundation (ITR 0426971), the NSF-NIRT grant, award number 0507023, and from our colleagues in MaSTRI, the Modeling and Simulation Technology Research Initiative at the University of Virginia.

AUTHOR BIOGRAPHIES

LINGJIA TANG is a Ph.D. Candidate in Computer Science and a member of MaSTRI at the University of Virginia. Her interests include optimization in coercion process. She earned her B.S. in Computer Science in Zhejiang University, China. Her email address is <lt8f@cs.virginia.edu>.

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. His email address is <reynolds@cs.virginia.edu>