Abstract. An emerging, yet extremely important topic in simulation application and research is simulation optimization. Emphasis in the literature, in professional conferences, and particularly in commercial software packages underscore the importance of simulation optimization. We address this topic in a more general, model optimization venue. In particular we describe the use of a genetic algorithm in a pure meta-heuristic role to search the space of model parameter settings to find a “best” set of parameter settings. We discuss three specific applications of a genetic algorithm meta-heuristic applied to a model optimization task.

1. Introduction

Models are used in all aspects of decision making, both in the public and private sectors. Models are simplifications of systems we use to gain knowledge and insight about the system. As M. G. Kendall noted, “[M]odels are for thinking with” [1]. Part of this requisite insight may be determining best, even optimal settings for the system. Simulation models are descriptive in nature whereas models that are used to suggest optimal or best settings are prescriptive. However, since simulation models can typically represent more complex systems than the general prescriptive model, we really want to “strap on” a prescriptive capability to our descriptive models. Simulation optimization via a meta-heuristic provides that strap-on.

The term meta-heuristic was coined by Glover [2] but our definition is:

A meta-heuristic is a master strategy that guides subordinate models to locate high-quality solutions. Further, a meta-heuristic may employ a single solution or a population of solutions.

Popular meta-heuristics are tabu search, scatter search, simulated annealing, and our current focus, genetic algorithms (GA). These heuristics are proven optimizers in their own right. To adapt a heuristic to a meta-heuristic role, one simply replaces the functional evaluation component with a call to the model of interest. The model output becomes the basis of the evaluation function. This provides “a cross-fertilization between discrete optimization heuristics and discrete event [simulation]” [3].

We do not restrict our attention to just simulation models. Meta-heuristic search strategies can guide any model. We wish to extend these strategies to other fields, particularly those involving military or military-related issues.

After some background information, we present three applications of a genetic algorithm meta-heuristic to military-related models. The first application addresses a multi-scenario optimization problem. The second application investigates the optimal allocation of air assets to specific missions. The third approach uses a GA to generate good solutions to aircraft repair time analysis problems.

2. Simulation Optimization Defined

Drawing analogy to Law [4], we define (simulation) model optimization as involving \( k \) decision (or design) variables, \( V_1, \ldots, V_k \) whose particular values, \( V_i=v_i, \ldots, V_k=v_k \), are evaluated yielding some measure of fitness, \( f(V_i=v_i, \ldots, V_k=v_k) \). The optimization problem is to find the vector \( V_i=v_i^*, \ldots, V_k=v_k^* \) such that \( f(V_i=v_i^*, \ldots, V_k=v_k^*) \) is a “best” value. Law and McComas’ tutorial [4] states, “…that optimization is the ‘hottest’ topic in discrete-event simulation today,” despite being “in its infancy.” We note the emphasis on the topic by WSC presenters such as [5], [6], and [7].
and by software packages such as Arena, Witness, Crystal Ball, and Frontline’s Premium Solver for Excel.

3. Genetic Algorithms
A GA draws an analogy to biology’s survival of the fittest. Natural selection reinforces characteristics most amenable to a species survival. Genes within chromosomes of the stronger members pass to subsequent generations through the reproduction process.

This selection paradigm fits optimization applications. Problem solutions (phenotypes) are encoded (genotypes), usually in binary format (genes). The set of solutions under consideration form a population with each solution considered a chromosome. The fitness of each member is generally the functional value of the phenotype, although specific applications may modify the fitness function, for example, to penalize problem constraint violations (penalty-based fitness function) in constrained optimization.

Fit chromosomes combine to produce chromosomes for subsequent populations. Member pairs of the population are selected for reproduction, usually based on some function of their individual fitness value. Genes from each parent are combined according to some predefined strategy to produce offspring (derivative chromosomes) and together these good parents and their offspring form a new, replacement, population.

The fundamental concept in GA optimization applications is that better solutions share “good” gene combinations, or schema, which carry over during the reproduction process. Over many generations, the better schema dominate and yield populations containing the best, possibly even the optimal solutions.

Although, there is no guarantee a GA will converge to an optimal solution, experience suggests GAs perform quite well.

4. Why Use Genetic Algorithms for this Application?
If a GA searches for and reinforces good schema how can that characteristic of a GA possibly apply to model optimization? In complex systems, there are many variables with complex inter-relationships. In such systems, the variable settings are the characteristics. We want those characteristic whose presence leads to strong solutions. We use the GA process to (hopefully) find those best variable combinations. We next consider three research case studies involving military-related models.

4.1 Multi-Scenario Force Structuring
Three important questions face the military planner. These are:
- How do we fight wars and conflicts?
- Who do we fight in these wars and conflicts?
- How do we equip our forces to fight these wars and conflicts?

Uncertainty underlies each of these questions. Despite the uncertainty involved the military planner must answer each of the above questions with respect to various planning scenarios. A real challenge facing analysts in the planning and programming function is how to reconcile the competing requirements and restrictions among myriad planning scenarios and then determine the best mix of resources to best meet future, uncertain requirements.

Models play an important function in force structure analyses. Given some scenario, models might provide as output a prescribed force structure necessary to attain the goals for that scenario. Conversely, given some defined force structure, models might provide a “best use of force” recommendation or goal attainment shortfalls based on the input.

The challenge is how to recommend a single force structure across diverse, competing scenarios. A force ideally suited for one scenario may be less than optimal for the other, and vice versa. Trade-off studies must reconcile the differences but how to conduct such a trade-off study is ambiguous. One approach to this force structure trade-off problem is to solve each scenario separately, giving the more “stressing” scenario its preference for weaponry, with the less stressing scenario picking among remaining resources. In reality, planners must also consider and plan the force to accommodate myriad other small-scale contingencies. We would like to evaluate the trade-offs among these many disparate scenarios in a systematic fashion, but have mostly settled for group-based reconciliation.

We define a robust force structure (solution) as that force structure (solution) “providing the best overall outcome as evaluated with respect to some set of scenarios each of which has an associated likelihood of occurrence.” We work directly in the multi-scenario space treating it as a composite of any component scenario spaces where each component space contributes relative to its likelihood of occurring or relative importance weight. We employ a GA to search this composite multi-scenario space for the “maximizing” force structure.

Our approach is presented graphically in Figure 1. Central to the approach is a CONTROLLER interface between the genetic algorithm and the combat model conducting the evaluations. The GA guides the search process providing the CONTROLLER potential solutions (input force structure) and receiving from the CONTROLLER evaluations of those solutions. The COMBAT MODEL receives a vector of input parameters (and a scenario) from the CONTROLLER, evaluates the input, and returns the evaluation. The CONTROLLER accepts the potential solutions, provides those to each of the scenario evaluators in the COMBAT MODEL, and combines all data into the final value or fitness of the potential solution used by the genetic algorithm. This process continues until predefined stopping conditions are satisfied at which time the best, or set of best, solutions are returned.
We applied this methodology using the Combat Forces Assessment Model (CFAM) model and three notional Air Force Air Expeditionary Force (AEF) scenarios [8]. Given some inputs (e.g., budget weapons, etc.), and CINC goals (e.g., time to halt an enemy), CFAM returns a prescribed mix of weapons and platforms to best meet the defined goals. Conversely, provided an input force structure, CFAM can provide information regarding the usefulness of that force structure within a defined threat scenario. We use the latter approach when using CFAM to evaluate potential solutions.

4.2 THUNDER Campaign Model Optimization

THUNDER, a U.S. Air Force campaign level model, is a two-sided model designed to simulate joint or combined forces conventional war at the theater level. While designed primarily to simulate air combat, it contains a ground combat module based on the Army’s Concepts Evaluation Model (CEM). THUNDER models all major operations in a theater -- air war, ground war, and resupply. Large combat simulation models such as the THUNDER are designed to simulate conventional war and rely on a vast amount of data in order for an analyst to conduct a study. This data comes from several sources that include:

- geographical / terrain characteristics,
- weapon system characteristics,
- engineering model estimates (e.g. probability of kill),
- intelligence estimates (e.g. number of enemy tanks),
- operational doctrine, and
- analyst judgment.

Most of the data is based on intelligence estimates or engineering data and not subject to analyst interpretation. However, several parameters settings require analyst judgment and may not be optimal. An example of this is the apportionment (percentages) of air power sorties to various mission categories. We use a GA to determine the optimal mission apportionment for THUNDER.

The air war models the mission planning sequence as shown in Figure 3. In THUNDER, aircraft can be apportioned to over 20 mission categories. An analyst determines the apportionment percentages of aircraft to mission categories. THUNDER uses a linear program to optimize aircraft and weapons to target allocation and generates air tasking orders based on the analyst’s apportionment [9].

We focused on 5 ground attack missions -- Strategic Target Interdiction (STI), Long range Interdiction (INT), Offense Counterair (OCA), Close Air Support (CAS), and Battlefield Air Interdiction (BAI). The ground attack missions are flown by multi-role aircraft while the remaining missions have a limited number of specialized aircraft that perform only one mission such as controller or electronic countermeasures (e.g. jamming) aircraft.

THUNDER provides several Measures of Effectiveness (MOE). The most common are Forward Line of Troop (FLOT) movement, enemy aircraft destroyed, strategic targets destroyed, sorties generated, and enemy armor
Three MOEs were chosen for this study. They are FLOT, percent enemy aircraft destroyed and percent strategic targets destroyed. FLOT was used as the overall fitness function with the other two MOEs set as minimum constraints. An additional constraint ensures the sum of the ground attack mission percentages sum to one. This is expressed as:

\[
\begin{align*}
\text{min} & \quad \text{FLOT depth} \\
\text{subject to} & \quad x_3 = 100 - x_1 - x_2 - x_4 \\
& \quad \% \text{ strategic targets destroyed} > 25% \\
& \quad \% \text{ enemy aircraft destroyed} > 40% \\
& \quad 1 < x_i < 100 \quad \text{for } i = 1, 2, 3, 4, 5 \\
\end{align*}
\]

where \( x_1 \) is the \% STI missions; \( x_2 \) is the \% INT missions; \( x_3 \) is the \% OCA missions; \( x_4 \) is the \% CAS missions; and \( x_5 \) is the \% BAI missions.

If any of the constraints were violated, a penalty was used instead of FLOT depth. A smaller penalty was used if the percent of strategic targets destroyed was between 20 and 25\% and the percent enemy aircraft destroyed was between 35 and 40\%.

Theater warfare limited to a 5-day battle of Red attacking into Blue territory was simulated and the MOEs at the end of the 5 days were collected using a process similar to Figure 1. The Results Curves for the GA versus an analyst’s judgment are shown in Figure 4. After only 5 generations, the GA improved over the analyst’s judgment and continued to find steady improvement until approximately generation 65 when the GA found its best solution.

**Figure 4:** Result Curves for THUNDER example

### 4.3 Repair Time Analysis Application

Traditionally, simulation has been based on a descriptive model with the focus on evaluating performance and drawing conclusions about what happens in the system. It allows analysts to study the results of their choices for the system but does not necessarily direct them towards changes to improve the system. Through detailed understanding of the dynamics and interactions of the system components, an expert can study a simulation and, based on criteria for success, decide to modify it or leave it undisturbed. Mollaghasemi and Evans [10] argue that traditional simulation is not enough to effectively analyze and optimize large, complex systems. Prescriptive models, on the other hand, are used to generate decisions, not analysis, about system design.

In optimizing a system, it is important to know not only the current performance, but also what changes lead to improved performance. Prescriptive models do this by working toward the system goals. Analysts need a prescriptive modeling tool to explore the solution space and find good, feasible designs and remove poor designs from consideration. Also, analysts using to interact with the simulated designs used a descriptive modeling method to study the effects of different scenarios and to develop trust in the designs’ strength and flexibility. To meet this two-pronged objective of descriptive and prescriptive modeling methods, a framework that integrates genetic algorithms with interactive simulations is built.

We developed an approach to integrate prescriptive and descriptive modeling methods to support repair time analysis problems. The approach was implemented in the Java programming language and consists of two components: a solution explorer and an interactive analyzer. The solution explorer uses a GA to explore the design space quickly and efficiently generating a final solution set that outperforms the original. With the interactive analyzer, the analyst visualizes and performs what-if analysis on alternative solutions to select a final solution. The results are overviewed below after we describe the application.

An airbase consists of different entities, including aircraft, mechanics, support equipment, and flight schedulers, that work together to achieve the overall requirements of a mission. Aircraft are assigned to specific sorties listed in the mission log based on the abilities of the aircraft and the goals of the mission. The aircraft take off, perform special mission-oriented tasks, land, and undergo systematic maintenance checks of their subsystems. Each aircraft is comprised of many subsystems which, during operational use, may fail. Subsystems can also experience different degrees of failure hence maintenance effort can vary. If there are no maintenance problems, the aircraft wait for their next scheduled take-off. Otherwise, the aircraft is sent to maintenance before it is readied for its next flight.

Logisticians must carefully consider the needs of completing a mission and allocate resources to fill those needs. Aircraft must be carefully assigned to complete the mission requirements of the base. Failures and routine maintenance must be dealt with efficiently and speedily to avoid mission aborts. Failure to accurately predict the correct amount of aircraft and maintenance resources can lead to costly resource excesses or shortages of resources needed to meet mission requirements. Critical maintenance resources include skilled mechanics, support equipment and facility space.

Analysts must have the access to relevant data, organized in an intelligent manner, which will help them make decisions on how, or if, to modify a system. Analysts need access to statistics to gauge system performance based on the performance of important indicators (e.g., the number of aborted missions and
cumulative maintenance wait time). With misleading or poorly organized data, analysts may reach the wrong conclusions about the system, leading to errors.

The particular scenario we studied involves the missions of the FX-99, a notional aircraft. Mechanics, equipment, hangars and spare parts are used to repair failures and conduct routine maintenance on the FX-99’s meager collection of 23 subsystems. The schedule of flights, failures, and even the maintenance behaviors are based on data files that can be manipulated to test different possibilities during analysis. Both the solution explorer and the interactive analyzer were evaluated. Details of the individual evaluations are provided in Schneider [11]. Results are summarized below.

In the evaluation of the solution explorer, eight design solutions served as the initial population for the solution explorer evaluation. For each generation, the designs were studied with different seed values and the generation mean was recorded. The Mann-Whitney test was based on data files that can be manipulated to test different possibilities during analysis. Both the solution explorer and the interactive analyzer were evaluated. Details of the individual evaluations are provided in Schneider [11]. Results are summarized below.

In the evaluation of the solution explorer, eight design solutions served as the initial population for the solution explorer evaluation. For each generation, the designs were studied with different seed values and the generation mean was recorded. The Mann-Whitney test demonstrated an improvement in the population mean (p > 0.002) very rapidly. Subsequent improvements in the designs took more generations. Although there was convergence after six generations, it was not clear how close the converged solution was to the optimal.

5. Discussion

Efforts to optimize simulations are by no means new. Statistical methods, such as designed experiments and the resulting response surface methodologies have worked quite well for a long time. However, when solution spaces are rugged, as found with most complex models, then systematic techniques such as response surface methodology may not provide the type of results desired.

Genetic algorithms, because they operate on the “characteristic space” of a problem, and can learn what types, levels, and combinations of parameters are found in the stronger solutions and are thus insensitive to solution topologies. Our research lends credence to the utility of meta-heuristics for model and simulation optimization.

However, the “no free lunch” crops up as it so often does and there really are some pressing research issues. First is technique scalability. By this we mean that though our results on smaller problems are promising, will these results scale appropriately when the technique is applied to larger problems?

Second there is complexity of the models and their inputs. More complex models require more parameters in any characterization. Will genetic algorithms continue to perform sufficiently? Convergence of a GA is partially a function of the size of the chromosomes, so problem growth may inhibit this techniques viability as more model evaluations are required to obtain answers.

Which meta-heuristic is best? We focused on genetic algorithms. However, it is quite possible one of the other popular heuristics, or some hybrid form of them, provide a better solution in a quicker amount of time by exploiting domain knowledge something not usually associated with a GA.

Finally, there is the question of what to do with all the data emanating from the optimization effort. Sensitivity analysis is well established in the classical optimization arena, even in simulation practices. However, this area has not been examined in the simulation optimization area. This might not seem important, but given the rugged solution landscapes being searched, knowledge of peaks and plateaus, or valleys, could be of tremendous use when trying to decide among competing, seemingly equal model parameterizations.

6. Concluding Remarks

We considered a genetic algorithm as a meta-heuristic “strap on” to various models with a military focus. The intent, as borrowed from the simulation optimization literature, is to locate those model parameter settings that provide a “best” set of conditions. Classical optimization methods have been used in these simulation optimization contexts but as more complex simulations are used, these classical methods will tend to break down in the face of the rugged solution landscape we are searching.

We consider three separate applications of model optimization using a genetic algorithm to guide the search and a particular model to evaluate potential solutions. Our results are optimistic but not without additional research challenges. While [4] views simulation optimization the hot topic in simulation, and we agree it should be, we cannot help but conclude that our promising results must be tempered with knowledge of the additional work ahead to fully realize the benefits of these optimization techniques.

References


