Generative Multisimulation: Decision-Support under Uncertainty using Evolutionary Multimodels

Levent Yilmaz, Bradley Mitchell
Auburn University, 3116 Shelby Center, Auburn, AL, 36849, USA
Corresponding author : yilmaz@auburn.edu

ABSTRACT
Generative Multisimulation (GMS) is a generative simulation methodology, which introduces a symbiotic adaptive decision support capability for systems with shifting, ill-defined, uncertain environments. Rather than rely on a single authoritative model, GMS explores an ensemble of plausible models, which are individually flawed but collectively provide more insight than would be possible otherwise. A case study based on a UAV team search and attack model is presented to illustrate the potential of GMS. Preliminary results demonstrate the potential of GMS to produce a large degree of exploratory behavior, followed by increased exploitative search behavior as the physical system unfolds.

KEYWORDS
Generative multisimulation, evolutionary multimodels, autonomic simulation, decision-support systems

1. Introduction

Strategy problems are typically characterized by significant uncertainty [Davis and Bigelow 1988]. Proper simulation-based decision support methodologies that facilitate making decisions in field settings could improve modeling Course of Actions (COAs), simulating them faster than real time, and then performing COA analysis to improve robustness and resilience of decisions. Exploring the effectiveness of alternative COAs requires dynamic updating, branching, and simultaneous execution of simulations, potentially at different levels of resolution. Dynamic updating of simulation models is a key requirement for using simulation as a tool to improve systems in which information only becomes available once the system is in progress [Yilmaz 2004;Yilmaz et al. 2007]. In these types of systems, the initial conditions provide little or no insight into how the system may develop over time. Emergent behavior that arises dynamically is a primary source of information in these systems. Exploiting this information to enable robust decision-making in a timely manner requires the ability to observe the system in real-time and adapt useful characteristics for the system with as little computational effort as possible.

The notion of self-organization provides a useful metaphor for the design and development of next generation simulation infrastructures. The term self-organization has been used in different areas with different meanings, as is cybernetics, thermodynamics, biology, mathematics, computing, and information theory. A system can be described as self-organizing if its elements interact to dynamically achieve a globally desired behavior. The behavior is not imposed and determined in a top-down manner, rather it is achieved autonomously as elements of the system adapt and evolve to changing environmental conditions. Similarly, autonomic computing paradigm relies on perception and

understanding of the environmental context to self-manage the computational infrastructure so that optimal operating conditions can be attained.

Symbiotic Simulation (S2) [Fujimoto et al., 2002] involves the use of simulation systems that are synchronized with the physical systems to enable mutually beneficial adaptation. In S2, simulation outputs are examined and used to determine how the physical system may be optimized. Similarly, measurements from the physical system are used to validate the simulation. When uncertainty in the physical system is present, multiple what-if simulation experiments can be helpful in adjusting the physical system. However, since the number of what-if experiments that may be performed is limited by both computational and real-time constraints, the ability to conduct an efficient search of the model space is essential.

Generative Multisimulation (GMS) is intended as a self-organizing generative S2 technique appropriate for physical systems characterized by distributed, dynamic and uncertain conditions. It is heavily inspired by the fields of Multisimulation [Yilmaz 2007], Exploratory Analysis [Davis and Bigelow 2000], and Exploratory Modeling [Bankes 1993], which involve the use of an ensemble of plausible models to provide insight in the absence of a single authoritative model. The salient feature of GMS is the use of evolutionary computation in terms of a Genetic Algorithm (GA) to evolve the model ensemble in response to changes in the physical system. With this feature, it is conjectured that an effective search of an uncertain model space would be possible, thus permitting synchronization with the physical system.

1.1 Motivation

Experimenting with evolutionary and/or contingency models in real-time on demand would be critical for decision support in unstructured problems with the characteristics of (1) deep uncertainty, (2) dynamic environments, and (3) shifting, ill-defined, and competing goals. The major challenges pertaining to decision-making in such asymmetric and irregular environments include the following [Yilmaz 2007]:

- For most realistic problems, the nature of the problem changes as the simulation unfolds. Initial parameters, as well as models can be irrelevant under emergent conditions. Relevant models need to be identified and instantiated to continue exploration. Manual exploration is not cost effective and realistic within a large problem state space.
- Our knowledge about the problem being studied may not be captured by any single model or experiment. Instead, the available knowledge is viewed as being contained in the collection of all possible modeling experiments that are plausible given what is known and what is learned.
- Dealing with uncertainty is paramount to analyzing complex evolving phenomena. Adaptivity in simulations and scenarios is necessary to deal with emergent conditions for evolving systems in a flexible manner.

1.2 Strategy: The Basis for GMS

Addressing the above challenges requires a simulation system to cope with an unpredictable environment autonomously through flexibility and robustness. Flexibility can be achieved using different but closely related approaches:

- **Adaptation**. The system changes its behavior to cope with the change through

learning and adaptation.

- **Anticipation**. An anticipatory system is a system whose next state depends on its current state as well as the current image(s) of its future state(s).

Coping with robustness and resiliency under changing conditions, on the other hand, requires capability to function in the face of perturbations. Proper mechanisms for dealing with uncertainty are abundant in nature. For instance, arthropod (insect) eyes which are called compound eyes, are made up of repeating units, the ommatidia, each of which functions as a separate visual receptor. Each ommatidium is pointed at just a single area in space and contributes information about only one small area in the field of view. The compound eye is excellent at detecting motion. As an object moves across the visual field, ommatidia are progressively turned on and off. Because of the resulting "flicker effect", insects respond far better to moving objects (e.g., situations) than stationary ones.

Similarly, when model excursions are viewed collectively, the behavior of plausible models can be informative despite the flaws of each individual model. Because of the presence of uncertainty, there may be many plausible models that could represent a system [Bankes 1998]. Similarly, knowledge of the system constrains the set of plausible models. Multisimulation, Exploratory Analysis [Davis and Bigelow 1999] and Exploratory Modeling experiment with ensembles of models as experimentation with a single plausible model would be just as likely deceptive as informative. Among a set of plausible models, variation occurs according to input uncertainty and structural uncertainty.

1.3 Decision-support under Uncertainty using GMS

GMS supports decision making by conducting a search through a potentially infinite space of models. The search works by evolving a set of potential system configurations for a dynamic set of environmental conditions. In this paper, we explore the following problems:

- What are different forms of uncertainty and how can they be modeled in such a way to facilitate exploration of alternatives and variation of configurations in an efficient manner?
- How can the key principles of CAS evolution be leveraged to design the evolution process of the decision support system so as to attain robustness under emergent order and unpredictable future state of the physical system.

Systems characterized by non-linear interactions among diverse agents often exhibit emergent behavior that may be very different from what the initial conditions of these systems would suggest. Traditional simulation techniques that rely on accurate knowledge of these conditions typically fail in these cases. The goal of GMS is to enable robust decision making in real-time for these problems. Rather than rely on a single authoritative model, GMS explores an ensemble of plausible models, which are individually flawed but collectively provide more insight than would be possible otherwise. The insights derived from the model ensemble are used to improve the performance of the system under study. Likewise, as the system develops, observations of emerging conditions can be used to improve exploration of the model ensemble. In essence, a useful co-evolution between the physical system and GMS occurs.

2. Generative Multisimulation: Methodology and Implementation

GMS behaves/evolves according to three key principles: order is emergent as opposed to predetermined [Zeigler 1989], the system's history is irreversible, and the system's future is often unpredictable [Holland 1975]. Agents are semi-autonomous units that seek to maximize some measure of goodness, or fitness, by evolving over time. Agents scan their environment and develop schema representing interpretive and action rules. Existing agent and model schema undergoes three types of change: first order change, where action is taken in order to adapt the observation to the existing schema; second order change, where there is purposeful change in the schema in order to better fit observations; and third order change, where a schema survives or dies because of the survival or death of its corresponding fitness. Schema changes through random mutation. Schema change generally has the effect of making the model ensembles more robust (it can perform in light of increasing variation or variety), more reliable (it can perform more predictably), or grow in requisite variety (in can adapt to a wider range of conditions). The fitness of the model ensemble is a complex aggregate of many factors, both local and global. The general health or fitness of the agents within a single model determines what the probability of change will be. Optimization of local fitness allows differentiation and novelty/diversity; global optimization enhances the coherence of GMS and induces long term memory. In general the probability of second order schema change is a nonlinear function of the fitness value.

2.1 Hybrid Exploration

In order to efficiently search a potentially infinite number of plausible models, GMS uses a hybrid exploration technique. As shown in Figure 1, uncontrollable inputs and controllable inputs are handled with an input analysis module and an output analysis module respectively. Measurements of the physical system's behavior are used to hypothesize distributions for uncontrollable inputs. As more details of the physical system's environment become known, the fidelity of these distributions with the actual values of the physical system should improve. Controllable input factors representing the configuration of the physical system are evolved using a genetic algorithm. A set of controllable inputs that completely describes a potential system configuration can be thought of as an individual. The controllable factors therefore, are considered to be the decision variables of a given problem while the uncontrollable factors determine the shape of a dynamic fitness landscape. An evolutionary algorithm enables the adaptation of individuals through a process of natural selection, with better performing individuals being more likely to survive and ultimately be used as the configuration settings for agents within the physical system itself.

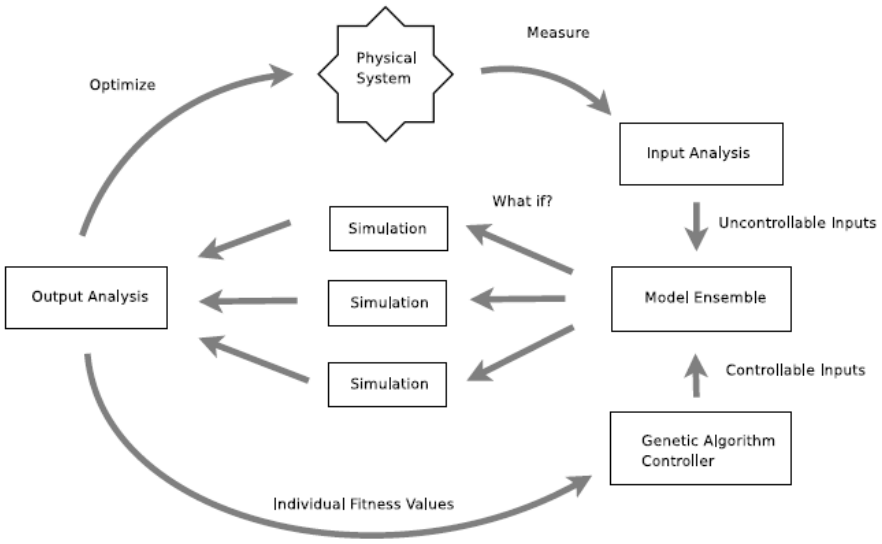


Figure 1: Symbiotic Simulation

2.2 Partial Model Ensembles

Once a distribution has been hypothesized for each uncontrollable factor in the physical system, and the associated parameters for those distributions have been estimated, each factor is sampled multiple times. These samples are stored in a 2-dimensional array referred to as a Partial Model Ensemble (PME). Variates for each sampled distribution are associated on a per-sample basis and placed into the same row of an array. Each column of the array stores variates from a single input factor, and each row of the array is considered to be a partial plausible model of the physical system. Later, these partial models are combined with potential system configurations to create fully specified plausible models for simulation. To hypothesize distributions for uncontrollable inputs and estimate their parameters, a means of observing the physical system is required. As real-time symbiotic simulation is performed, observations of the physical system enable more accurate estimates of the input distribution parameters. With these improvements, the space of plausible models shrinks, allowing the system to simulate fewer models in greater detail.

2.3 Combined Model Ensembles

In GMS, a Combined Model Ensemble (CME) is a specification for conducting a series of simulation experiments involving a single individual from a population of potential system configurations. The goal of these experiments is to test an individual in multiple possible environments. Each simulation experiment examines the individual in the context of a set of uncontrollable factors representing a single possible environment. The fitness of the individual for a particular environment is obtained as an output from a simulation experiment. Resulting fitness values from all experiments with the individual in each respective environment are then averaged together to obtain an overall fitness for the individual, which is used to determine its probability for reproduction and survival within the genetic search.

The layout of a CME is shown in Table 1. The possible environments with which the individual is to be tested are determined by a PME of Uncontrollable Factors making up

the left hand side of the table. Copies of the individual are paired with each row of the PME. A single row of the CME thus includes both a sampled set of values from the uncontrollable factor distributions and a copy of the individual which specifies the settings of the controllable model factors. Taken together, these two sets of elements form a single plausible model described by the entire row of values in the CME.

Table 1: Combined Model Ensemble- A Combined Model Ensemble of *n* models is composed of a Partial Model Ensemble of *m* sampled variables, *X*, and an individual of *l* genes, *G*. Each row represents a single model for which one or more simulation replications should be performed. Note that the same individual is used in each model.

Uncontrollable Factors				Controllable Factors			
X_{11}	X_{12}	$\cdots$	X_{1m}	G_1	G_2	$\cdots$	G_ℓ
X_{21}	X_{22}	$\cdots$	X_{2m}	G_1	G_2	$\cdots$	G_ℓ
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
X_{n1}	X_{n2}	$\cdots$	X_{nm}	G_1	G_2	$\cdots$	G_ℓ

Note that for a given CME, the same individual is paired with each row of the PME. Furthermore, the same PME is combined with each individual in the population to create a set of unique CMEs, one for each individual. Since the same PME is used in each CME, each individual is evaluated using the same environmental conditions. In essence, the uncontrollable factor settings of the PME, which change each time a new PME is created, become a dynamic fitness landscape.

2.4 Local Adaptation: Particle Swarm Optimizer

Self-organization is a kind of aggregate behavior that is often associated with Complex Adaptive Systems (CAS). Self-organization as a property of an engineered system has been described as being one in which individual agents or units respond to local stimuli to achieve through a division of labor the efficient performance of some task. The collective efficiency of task performance must be greater than what could be accomplished individually. Additionally, self-organization involves the creation of an equilibrium state that arises from the local interactions of agents. This equilibrium may be achieved through either competition or cooperation.

In GMS, Particle Swarm Optimization (PSO) is used for implementing the learning element of adaptive agents. A broad introduction of PSO can be found in [Kennedy and Eberhart 2001]. We use PSO as a continuous numeric optimization technique in which a potential solution to a problem is characterized as a point in some n-dimensional space, with the number of dimensions being equal to the number of decision variables. As the name suggests, PSO uses a population of potential solutions. These solutions ``fly'' through the problem space over time. As each particle moves through the problem space, it records the best solution, *pbest* that it has found so far as well as the best solution, *gbest* discovered by the other members of the swarm. Particles tend to gravitate toward these two positions over time as they search for better solutions [Bratton and Kennedy 2001].

2.5 Global Adaptation: Genetic Search of Potential System Configurations

GMS evolves potential system configurations for use within the physical system. Unlike the creation of a PME, however, a search of the space of potential system configurations

requires the exploitation of a different type of system information, namely the performance characteristics of the configuration. Evolutionary Algorithms exploit this type of information to improve a search as they are well suited for optimization problems. Among Evolutionary Algorithms, Genetic Algorithms have a feature useful to GMS, which is the distinction between genotypic and phenotypic representation. A genetic encoding of the characteristics of a potential system configuration allows the same system to be interpreted in different ways and in varying levels of detail.

The structure of a typical GA is shown in Algorithm 1. An initial population of *m* individuals is randomly created and the fitness values of the resulting individuals are evaluated. The algorithm then enters a loop in which successive generations of individuals are evolved. An inner loop continues until a group of *n* children are created. Two individuals are chosen to be parents. Crossover combines the characteristics of both parents to create a new child. A mutation operator is then applied which may modify the child with characteristics not possessed by either parent. The fitness of the new child is then evaluated.

Algorithm 1 : A genetic algorithm.

```
1: pop[m] ⇐ createInitialPopulation( )
2: for i = 1 to m do
3:         pop[i] ⇐ calculateFitness( pop[i] )
4: end for
5: repeat
6:         children[n]
7:         for i = 1 to n do
8:                 parents[2] ⇐ selectParents( pop )
9:                 children[i] ⇐ crossover( parents )
10:                children[i] ⇐ mutatation( children[i] )
11:                children[i] ⇐ calculateFitness( children[i] )
12:        end for
13:        pop[m] ⇐ selectSurvivors( pop, children )
14: until termination = true
```

Once the children have been created, the total population of individuals is then reduced back to *m*. Depending on the design of the GA, survivor selection may or may not involve direct competition between parents and children. The evolutionary process continues until some termination condition (such as a fixed number of fitness evaluations) is reached.

2.6 GMS Component Architecture

GMS implementation, designed to operate in mission critical environments, is based on an independent component architecture in which the individual components of the system could execute in parallel and communicate via message passing [Braude 2004]. This could be especially useful for entities operating in the physical environment that would likely not have the hardware resources to process simulations. Rather, these entities would only need some means of measuring the physical system and sending those measurements to the components running the simulations. Figure 2 illustrates the essential components of the GMS system. Observations from the physical system are passed to an Input Exploration Component (IEC) responsible for conducting input analysis and selecting appropriate distributions for the uncontrollable model factors. Samples from these distributions are then used to create a PME, a copy of which is

integrated with each individual to form one CME for each Agent-based Simulation Process (ASP).

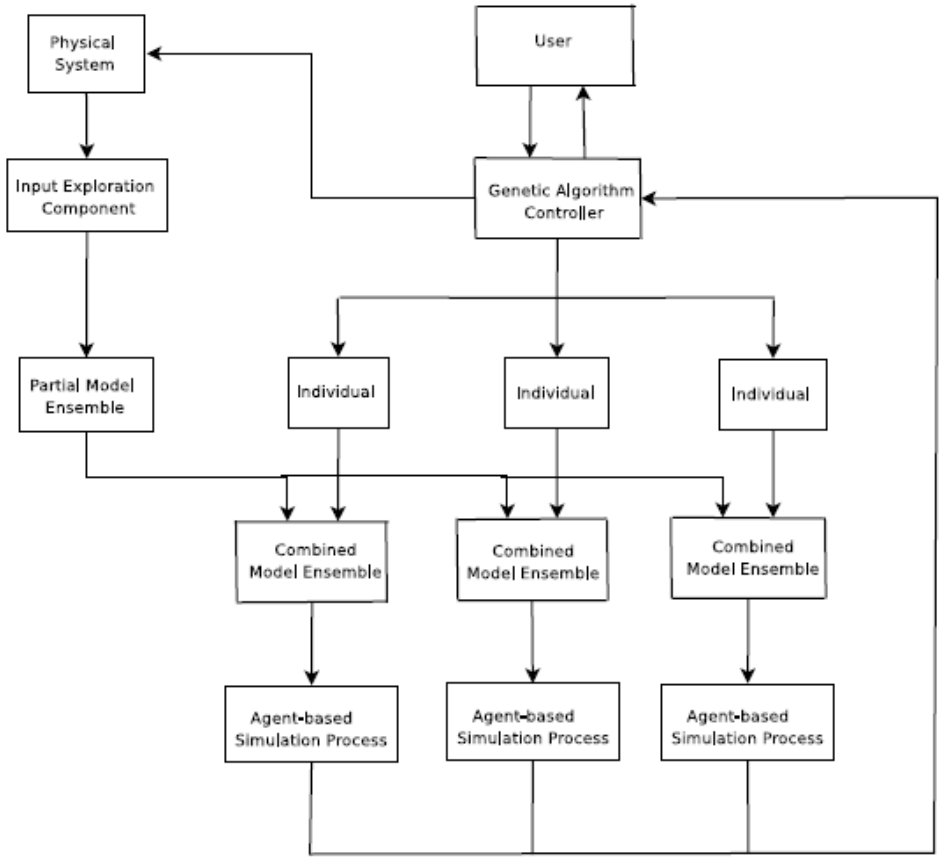


Figure 2: Components of the GMS Framework

The Genetic Algorithm Controller (GAC) is responsible for evolving the population of individuals which are used to form the CMEs. and a number of Agent Simulation Instances (ASPs) to simulate CMEs in parallel. Ideally, each ASP is mapped to one or more CPU cores. If the population used by the GAC is large, or if hardware resources are limited, multiple ASPs can run on a single core. If there is an excess of hardware, a GMS implementation should be capable of offloading models within the CME to multiple CPU cores. Outputs from the ASPs take the form of objective fitness values of the individuals that have been averaged across all of the replications specified by the CME. These are passed to the GAC which then assigns the fitness values to the simulated individuals before continuing execution of the GA. Ideally, it should be possible for a user to interact with the GAC in real time to examine the individuals generated and possibly seed new configurations to the population.

The sequence of operations that occur within the main execution loop of a GMS implementation are shown in a UML sequence diagram in Figure 3. Active components and processes, displayed as boxes across the top of the diagram include the Genetic Algorithm Controller, Agent-based Simulation Processes (of which several run simultaneously), an Input Exploration Component and the Physical System. An initial message is passed from the GAC to the IEC requesting a new Partial Model Ensemble. The IEC takes observations from the physical system and uses them to produce uncontrollable factor settings which are incorporated into the PME. The PME is then

passed to the GAC which uses them to create a CME for each ASP.

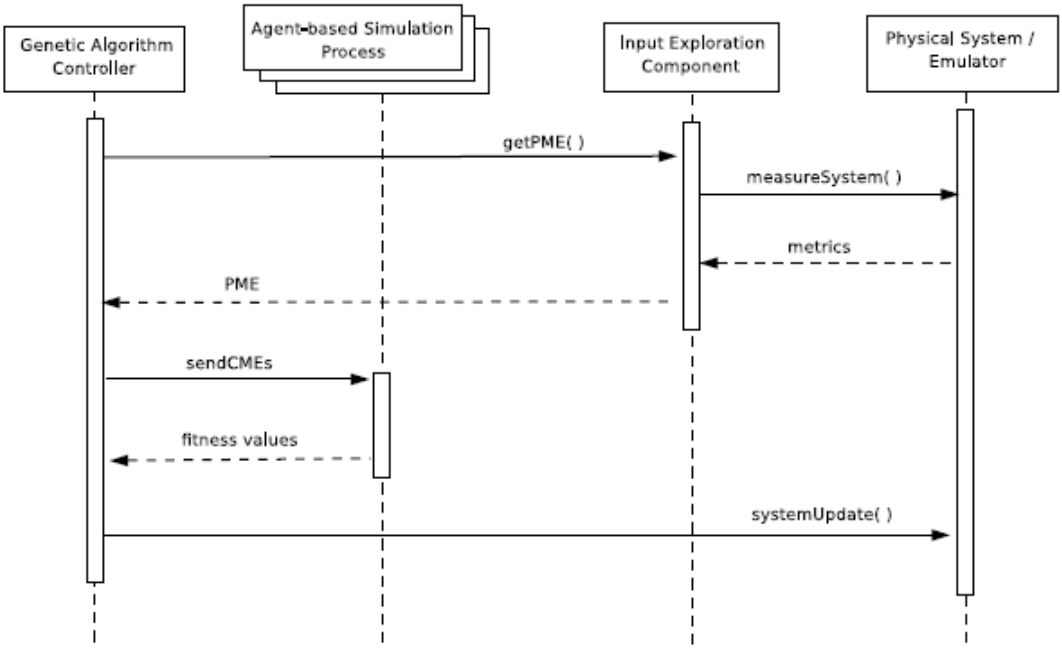


Figure 3: Component Interaction in GMS

When created, the CMEs are passed to the ASPs and simulated in parallel. Simulation results are examined on a per CME basis so that fitness can be assigned to a given individual based on its performance against the environmental settings specified in the PME. After fitness values have been assigned to individuals, the GAC evolves the population. Individuals with higher fitness are given preference for producing offspring, which are created using genetic operators of crossover and mutation. Due to the need to maintain a naturally parallel structure for the algorithm, large numbers of children should be generated before selecting survivors.

In addition to evolving the existing population, the GAC also selects the most fit individual of each generation to update the physical system's configuration. This continual process of dynamically updating the physical system helps ensure that the physical system is responding correctly to observed changes in the environment.

3. Case Study: UAV Search and Attack Scenario

To perform an assessment of the potential of the GMS methodology, a model based on the features of Complex Adaptive Systems was integrated into the parallel application. The field of autonomous UAV cooperation provides a natural environment from which to create such a model. Toward this end, an agent-based model of an autonomous UAV team in a Search and Attack mission was developed. The UAVs in this model interact through local communication only. There is no global system of coordination or prior intelligence of targets. A rule-based approach for UAV movement is implemented. This set of movement rules, which is based on general knowledge of the problem domain, results in a robust performance element for UAVs capable of both independent and cooperative actions. These rules are implementations of the steering behaviors described in [Reynolds 1999]. Similar implementations of these steering behaviors have been featured in a number of autonomous UAV studies including [Price 2006] and [Crowther 2004].

The Search and Attack scenario takes place in a 2-dimensional space of equal dimensions. UAVs begin play from a base located at the center of the map. Targets are distributed randomly across the map and their positions are initially unknown to the UAVs. Once they are launched, the UAVs must find and destroy all of the targets as quickly as possible. Figure 4 depicts the opening simulation steps in which UAVs are in the process of launching from the base (represented by a gray hexagon) and sweeping the map. Targets (represented by gray triangles) within the sensor envelope of a UAV have a chance of being discovered while those outside remain hidden.

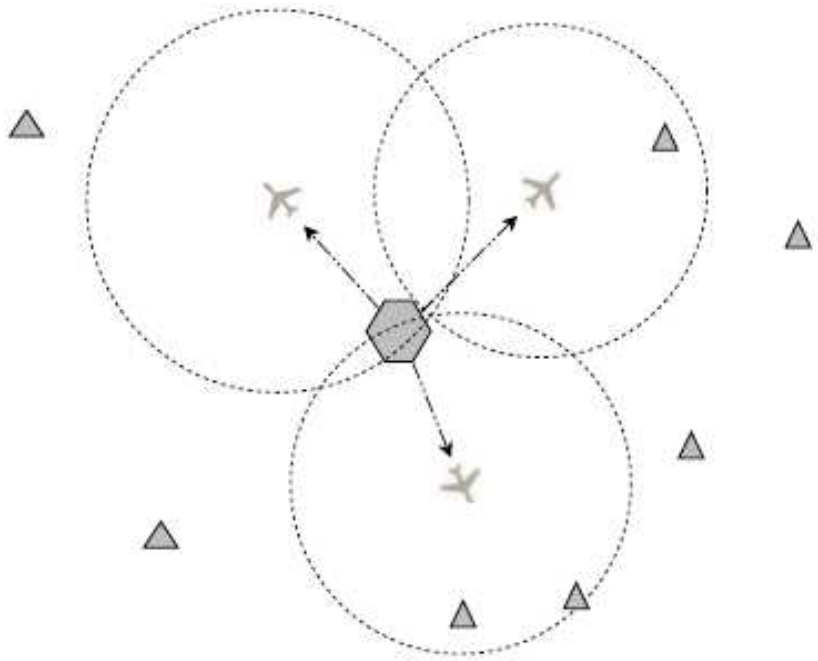


Figure 4: The Search and Attack scenario shortly after simulation start. UAVs depart from the centrally located airbase (represented by a hexagon with uniformly random initial movement vectors. The sensor envelopes of individual UAVs are represented by dotted circles.

Progression of the model is simulated through time-stepped execution, dividing simulation time into a number of equal size increments. During a simulation step, each UAV may act. The results of these actions are updated synchronously and may affect actions performed by other UAVs in the same time step.

4. Experiments with the Parallel GMS System

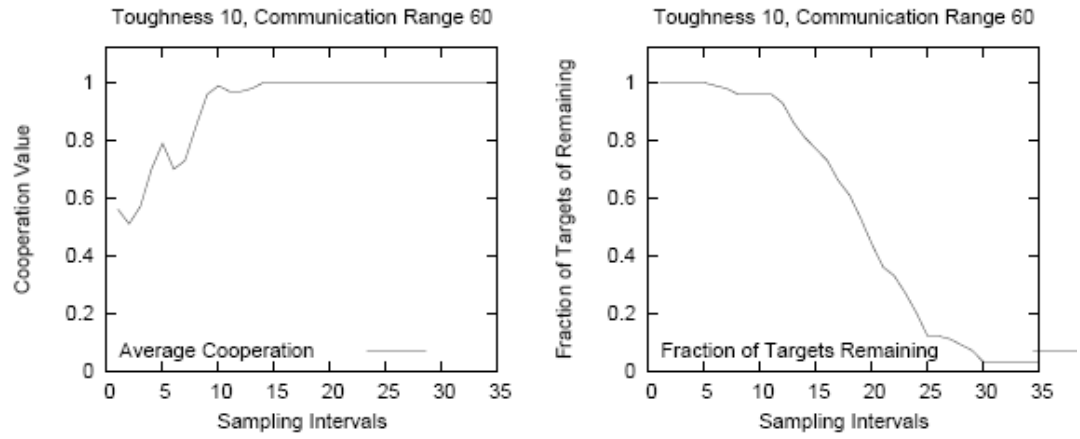
The goal of computational experimentation for this thesis was to provide an initial understanding of the potential of GMS as an S2 methodology. Toward this end, it was hoped that successful experimentation would help to answer a central question:

- Can the use of symbiotic simulation through ensembles of plausible models improve a physical system's performance?}

4.1 Establishing Face Validity

One interesting phenomenon observed in some of the experiments is that the behavior of the UAVs' average Cooperation values due to the influence of the distributed PSO. Figure

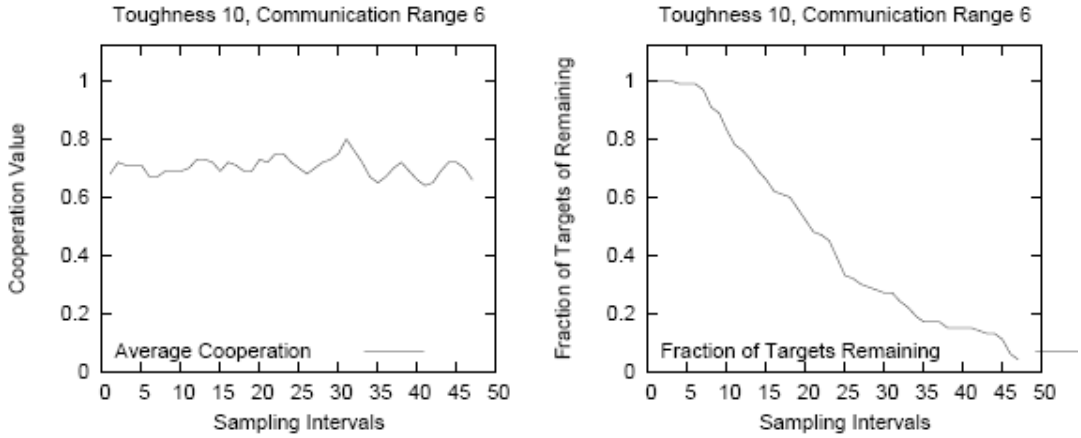
5 illustrates a single replication from one set of experiments, which was not included in the original experimental design, but appears instructive. This case involved a high Communication Range setting of 60 and one leader UAV. As discussed earlier, the distributed PSO evaluates the PSO objective function and updates its decision variables (including Cooperation and Base Distance) once per 50 time steps. This period is referred to as the sampling interval, shown across the bottom of the plot. Thus, at sampling interval 10, 500 time steps have elapsed in the simulation. This plot tracks the change in the average of all Cooperation values within the UAV team across a single model replication. In this case, by about the 10th sampling interval, the distributed PSO had evolved the team away from cooperative behavior.



(a) Change in Average Cooperation (b) Change in Target Population

Figure 5: Effect of Cooperation on Target Population with High Communication Range

The behavior shown in Figure 5 contrasts with what occurs when Communication Range is set to a low value. Figure 6 depicts that when Communication Range is set to 6, cooperation is more likely to occur. In this case, the average value of the PSO Cooperation value stayed relatively close to 0.5. The transitioning behavior to equilibrium in both cases is typical of self-organizing systems and helped us instill confidence in the learning mechanism.



(a) Change in Average Cooperation (b) Change in Target Population

Figure 6: Effect of Cooperation on Target Population with Low Communication Range

4.2 Emulator Objective and Individual Fitness

Experimentation with the Parallel GMS Application involved the use of a system emulator, rather than an actual physical system. This emulator is itself a simulation with the same structural model of autonomous UAV behavior used by the ASPs. It only differs in terms of the values used for its input factors and in that its controllable factor settings are not fixed for the duration of its execution. Similar to the model replications used in the stand-alone mode for face validity, the performance objective of the Emulator is to minimize the number of time steps required to eliminate all targets. Therefore, to synchronize the performance of the simulations executed in the ASPs with that of the emulator, the fitness of an individual is defined to be the average number of time steps required to eliminate all targets across all replications specified by that individual's CME.

The emulator is an ordinary UAV model simulation which receives controllable factor updates from GMS. Therefore, it was possible to examine the performance of the Parallel GMS Application by running an emulator with identical uncontrollable factor settings used in the stand-alone mode. Since the emulator and stand-alone models use the same model structure, their performance can be compared as long their uncontrollable factors and fixed model settings are identical. Our position is that the emulator's system would have an advantage over the system in a stand alone model. The controllable factors of Leader flags and Cooperation Thresholds can be modified by GMS to the benefit of the emulator, whereas the system in the stand alone model has controllable factor settings that are determined in advance and fixed for the duration of its execution. In order to compare the performance of the emulator with GMS against a stand-alone model, all fixed model settings such as Weapon Range, Average Weapon Effect, Weapon Effect Standard Deviation, Average Target Distance, and Target Standard Deviation were set to the identical settings used in the experiments described in the previous, as these settings were held constant across all treatments in that group. Furthermore, a subset of the treatments corresponding to one pair of settings of the uncontrollable factors is selected.

4.3 GA Design

One intent with GMS is for the Genetic Algorithm Controller to produce a large degree of exploratory behavior initially, followed by increased exploitative search behavior as the physical system changes. With this in mind, proportional selection was chosen as a parent selection operator for these experiments. The variation operators used for these experiments included one-point crossover, which obtains a single cut point that determines how the chromosomes will be divided and recombined to produce children. Each individual represented 20 UAVs and a non-overlapping survival model was selected so that all children survive while parents automatically die. This type of GA, which balances a relatively low selection pressure with low rates of variation from one-point crossover and low mutation, is suitable for encouraging exploration of controllable factors early in the physical system's development.

4.4 Emulator Results

The major limitation of these experiments was that the IEC was not yet implemented. Therefore, certain assumptions were made regarding emulator / physical system measurement and PME generation. The work of the IEC was simulated within the experiments. At run-time, the simulated IEC is initialized with uniform random distributions for three uncontrollable factors: Toughness, Communication Range, and Target Max Visibility. Note that Target Max Visibility was held constant across all

treatments. The true values used for these factors within the emulator were Toughness 10, Communication Range 6, and Maximum Target Visibility 5.

The parameterized distributions within the simulated IEC for the uncontrollable factors are arbitrarily selected to be uniform distributions of length 30 about each factor's true setting (truncated by zero as a minimum parameter). This included U(0, 25), for Toughness, U(0, 20) for Maximum Target Visibility, and U(0, 21) for Communication Range. At each iteration, a PME of 30 rows and 3 columns (one for each uncontrollable factor) is created. When combined with an individual, each CME therefore had 30 rows and 43 columns (3 columns for the uncontrollable factors and 40 columns representing 2 genes for each of the 20 UAVs in an individual). Each row, representing a single parameterized model, runs for one replication resulting in 30 replications for each CME. A population of 20 individuals mapped to 20 ASPs was used. This resulted in 600 model runs during each iteration of the main loop. The emulator replications included on average, 16 generations. Thus, on average, approximately 9600 model replications are performed by the ASPs for each replication of the emulator. More replications for each row of a CME and a larger population would have been desirable but were not used due to project time constraints.

For the experiment, 30 replications of the GMS Parallel Application on the Altix Supercomputer are run. As can be seen in Table 2, the performance improvement of the emulator is noticeable in terms of the number of time steps for mission completion compared to the best performing UAV team configuration among the stand-alone model tests.

Table 2: The results of the Parallel Application compared to the best performing Stand-alone Model. The Parallel Application provides significantly improved performance on average, but with higher variance.

Experiment	Average Time Steps	Standard Dev.
Emulator with SAMS	1597.5	164.34
Stand-alone model	1777.24	129.68

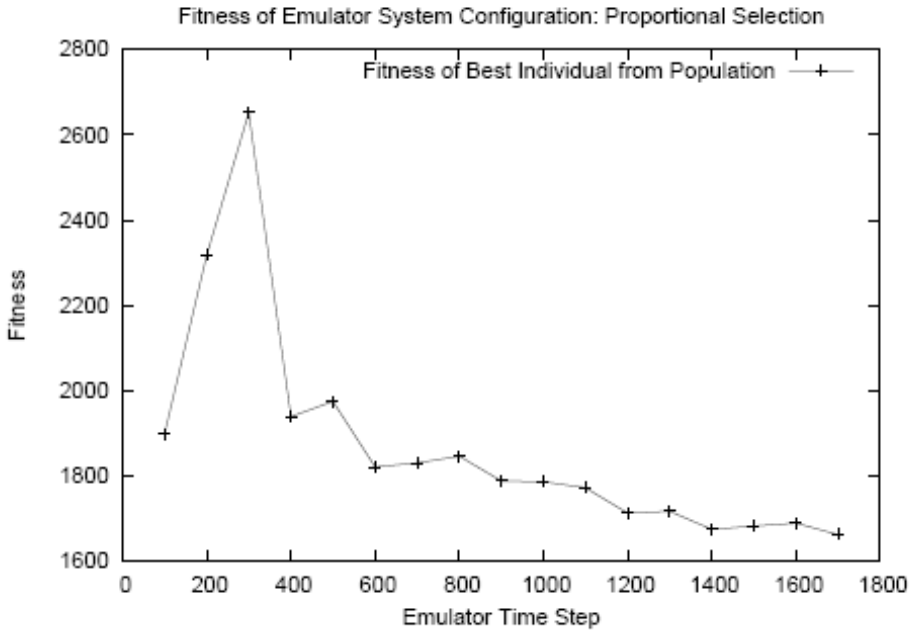


Figure 7: Fitness of Emulator System Configuration

A possible explanation for the increased standard deviation in GMS may be that the number of replications performed for each CME were simply insufficient to properly assess the fitness of the CME's respective individual. This might also explain the initial decrease in fitness in Figure 7 as the initial iterations of the main loop involve significant uncertainty in the uncontrollable factors. While these experiments seem encouraging, more experimentation is needed to explore this issue in detail. In particular, experiments with larger numbers of model replications need to be performed. Secondly, the possibilities for modification of Genetic Algorithm used by GMS have barely been scratched. Much research is needed to identify what design features of a GA (or EA) contribute most to improved performance.

5. Conclusions

This research examines the need for dynamic model updating for Symbiotic Simulation of systems involving interacting agents with complex, non-linear behavior. Our position is that these systems may not be effectively studied with traditional simulation techniques that rely on valid, authoritative models of the physical system. Instead, techniques such as Multisimulation and Exploratory Analysis, which experiment with an ensemble of plausible models are developed to deal with these problems.

The proposed Symbiotic Adaptive Multisimulation approach involves a Hybrid Exploration strategy to study an ensemble of plausible models. When parameterized to account for input uncertainty in controllable and uncontrollable factors, GMS is able to dynamically update a system emulator resulting in improved performance. This benefit is realized with the help of a Genetic Algorithm that evolves potential system configurations over the lifetime of the system emulator and which can be used to update the emulator. These updates consist of adaptive strategies that are passed on to the agents operating within the emulator. This initial study of GMS as a simulation methodology has shown encouraging results, but has also left many problems unsolved. In particular, further experimentation with larger numbers of model replications is required. Also, experimentation with additional GA designs can be performed to understand the features

that make an Evolutionary Algorithm suitable for GMS. The understanding gained could provide further improvements to the methodology in terms of speed and robustness. Other significant problems that remain are the incorporation of appropriate Multiresolution Modeling, input analysis for estimating uncontrollable factor distributions, and handling of structural uncertainty. Given these challenges, Autonomic Multisimulation appears to be a rich opportunity for further study.

REFERENCES

[Bankes 1993] Bankes, S. Exploratory modeling for policy analysis. *Operations Research,* 1993. Vol. 43, No. 1, pp. 435–449.

[Bankes 1998] Bankes, S. Policy analysis for complex and uncertain systems through computational experiments. In Proceedings of the *1998 IEEE Aerospace Conference*, 1998. pp. 125–132.

[Bratton and Kennedy 2007] Bratton, D. and J. Kennedy. Defining a standard for particle swarm optimization. *In Proceedings of the 2007 IEEE Swarm Intelligence Symposium*. Honolulu, HI: IEEE, 2007, pp. 120–127.

[Braude 2004] Braude, J. E. *Software Design: From Programming to Architecture*. John Wiley & Sons, 2004.

[Crowther 2004] Crowther, B. Flocking of autonomous unmanned air vehicles. *Aeronautical Journal*, 2004. Vol. 107, No. 10, pp. 111–124.

[Davis and Bigelow 1988] Davis, K. P. and J. H. Bigelow. The role of uncertainty in assessing the NATO-PACT central region balance," RAND N-2839. The RAND Corporation, Santa Monica, CA, 1988.

[Davis and Bigelow 1999] Davis, K. P. and J. H. Bigelow. Experiments in multiresolution modeling (mrm). *RAND Technical Report*, 1999.

[Davis and Bigelow 2000] Davis, K. P. and J. H. Bigelow. Exploratory analysis enabled by multiresolution, multiperspective modeling. In Proceedings of the *2000 Winter Simulation Conference*, 2000. pp. 127–134.

[Fujimoto et al. 2002] Fujimoto, M. R., D. Lunceford, E. Page, and A. M. Uhrmacher. *Grand challenges for modeling and simulation*. Dagstuhl report," 2002.

[Holland 1975] Holland H. J. *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI: The University of Michigan Press, 1975.

[Kennedy and Eberhart 2001] Kennedy, J and Eberhart C. R. C. *Swarm Intelligence*. San Francisco, CA: Morgan Kaufmann Publishers, 2001.

[Price 2006] Price, C. I. *Evolving self-organized behavior for homogeneous and heterogeneous uav or ucav swarms*. Master's thesis, Air Force Institute of Technology, Wright-Patterson Air Force Base, OH, 2006.

[Reynolds 1999] Reynolds, C. Steering behaviors for autonomous characters. (Retrieved on October 31, 2008 from http://www.red3d.com/cwr/papers/1999/gdc99steer.html), 1999.

[Yilmaz 2004] Yilmaz, L. Dynamic model updating in simulation with multimodels: A taxonomy and a generic agent-based architecture. In Proceedings of *SCSC 2004 - Summer Computer Simulation Conference*, pp. 3–8, 2004.

[Yilmaz 2007] Yilmaz, L. Toward next generation simulation-based computational tools for conflict and peace studies," *Social Science Computer Review*, 2007. Vol. 25, No. 1, pp. 48–60.

[Yilmaz et al. 2007] Yilmaz, L., A. Lim, S. Bowen, and T. Oren, Requirements and design principles for multisimulation with multiresolution, multistage models," In Proceedings of the *2007 IEEE/ACM Winter Simulation Conference*, pp.823-832.

[Zeigler 1989] Zeigler, P. B. Discrete event abstraction: An emerging paradigm for modeling complex adaptive systems. In *Perspectives on Adaptation in Natural and Artificial Systems, Essays in Honor of John Holland*, Sante Fe Institute, Oxford University Press, England, 1989.