Proceedings of the 1997 Winter Simulation Conference
ed. S. Andradóttir, K. J. Healy, D. H. Withers, and B. L. Nelson

SIMULATION-BASED PLANNING FOR MULTI-AGENT ENVIRONMENTS

Jin Joo Lee

LSI Logic Corporation
1625 McCarthy Boulevard
Milpitas, California 95035, U.S.A.

Paul A. Fishwick

Department of Computer & Information Science & Engineering
University of Florida
Gainesville, Florida 32611, U.S.A.

ABSTRACT

One of the key issues in reasoning with multiple interacting intelligent agents is how to model and code the decision making process of the agents. In Artificial Intelligence (AI), the major focus has been on modeling individual intelligence and a common approach has been to use operator or rule-based models to represent the decision making intelligence of an agent. If the purpose of the simulation is to precisely emulate a particular agent's intelligence, then such rule-based models may often be most appropriate. However, when the goal is to win the engagement in the battlefield, where the overall outcome may depend on individual execution of each task, the level of detail must be extended to the level of simulating individual task execution. In these cases, we have created a methodology, Simulation-Based Planning (SBP), that embeds one simulation inside another. The embedded simulation simulates the actions of agents *before* committing to a plan so that it may evaluate the desiredness of the actions. Plan alternatives are generated based on discrete paths through spatial regions of a domain, while specific optimal plans are generated through the use of experimental design and simulation. We have found that, through simulation-based planning, near-optimal plans can be selected by using simulation, in addition to using simulation once a plan has been adopted.

1 INTRODUCTION

In a multi-agent system, one needs to simulate the individual agents, but also the coordination of the agents and methods employed for achieving their goals through interaction, cooperation and adversity. Our approach is not to create cognitive models for individual agents, and then to apply simulation, but instead to create models for multi-agent systems that are tightly constrained by their environments. Example constraints include 1) geometric paths that are to be followed, 2) a task or mission to be accomplished, and 3) a strict set of operating conditions for all agents. With these constraints, it is feasible and logical to apply quantitative simulation techniques to support decision making involving multiple agents. Thus, our research focuses on the *constrained scenario* theme. In executing these scenarios, we create a simulation-based decision *model* which can be said to reside in a decision-making object or agent.

Related work was recently done by Dean et al. (**?**) which focuses on a method based on the theory of Markov decision processes for efficient planning in stochastic domains. Work by Wellman (**?**; **?**) is also related in that it also attempts to solve planning problems in uncertain environments. These approaches differ mainly because the state transitions in their models are deterministic (although probabilistic) whereas we can also model nondeterminsitic state transitions. There has been previous work done in the integration of AI and Simulation (**?**; **?**). However, combining different modeling paradigms and techniques is often a difficult task. Because of the ability to combine different modeling paradigms at multiple levels, we consider the object-oriented approach of multimodeling (**?**; **?**) to be a natural approach to solving the problem. Models that are composed of other models, in a network or graph, are called multimodels (**?**; **?**; **?**; **?**). Multimodels allow the modeling of large scale systems at varying levels of *abstraction*. They combine the expressive power of several well known modeling types such as FSAs, Petri nets, block models, differential equations, and queuing models. By using well known models and the principle of orthogonality, we avoid creating a *new* modeling system with a unique syntax.

2 SIMULATION-BASED PLANNING

SBP refers to the use of computer simulation to aid in the decision making process. In the way that game trees are employed for determining the best course

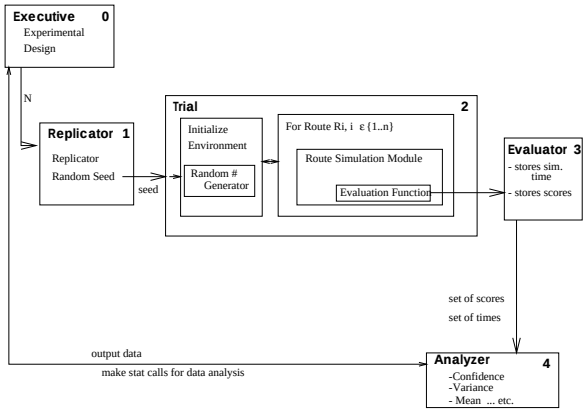

Figure 1: Generic Top Level Architecture of a Route Planner

of action in board games (using a game tree), SBP uses the same basic iterative approach with the following items: 1) a model of an action is executed to determine the efficiency of an input or control decision, and 2) different models are employed at different abstraction levels depending on the amount of time remaining for the planner to produce decisions. The military has been using simulation-based planning for many decades in the form of *constructive model simulation*. Our extension in SBP is one where we permit many levels of abstraction for a model, not just the aggregate abstraction level characterized by Lanchester equations and combat result tables. The idea is to allow the planner the flexibility to balance the need between the required level of detail and the amount of time given to make a decision. This is similar to Dean and Boddy's *anytime* algorithm (**?**). The notion that simulation can be used for decision making is covered in several disciplines, such as for discrete event based models (**?**).

2.1 The SBP Framework

We present the SBP framework in terms of three components: the simulation component, the experimental design component and the output analysis component. The model in Figure 1 is the top level general framework for simulation-based route planning systems.

2.2 Simulation Block : Trial

To use simulation in planning, we first need to identify the set of *controllable* and *uncontrollable* variables. Speeds, routes, actions of objects are controllable, whereas any kind of uncertainty such as uncertainty of weather conditions and outcome of combat are uncontrollable. The main objective of plan simulation is to gather the effects of the uncontrollable through repeated sampling (replication) while varying the parameters to find a near-optimal combination of controllable values in spite of the uncertainty. We say near-optimal because we can never guarantee the optimality of a plan given the uncertainties of actual plan execution. Unlike most plan evaluation schemes where the predicted state transition occurs by analytic (therefore, deterministic) and probabilistic functions, SBP nondeterministically chooses the "most likely to happen" transition given the situation by either sampling the probability or by executing a detailed model of the transition itself. Thus, both an advantage and possibly a drawback, SBP may come to evaluate transitions which do not have the highest probability of occurring.

Here we present some definitions that will help us formally describe the simulation algorithm we have created. Let $W = \{W_1, W_2, ..., W_k\}$ be the set of all objects in the environment and let $Q(t) = q_1(t) \times q_2(t) \times \cdots \times q_k(t)$ be the world state at time t, where $q_i(t)$ is the state of object W_i at time t. Also, we define $A_i(t)$ as the set of actions the object W_i can take at time t. Note that zero or more actions may be chosen from this set to be simulated at time t. The total number of routes is N and R_j denotes the jth route where $1 \leq j \leq N$. Then, the simulation algorithm is as follows:

```
INITIALIZE environment
(for every replication, setup all objects)
 For each route $R_{j}$
  WHILE ( (not success) or (not failure) )
   Obtain Q(t) FOR each object Wi
   Determine actions for Wi using set Ai(t)
   Update state qi(t) to qi(t+1) by either
    1) simulation of chosen actions
    or 2) sampling distribution
   Update world state Q(t+1) t = t + 1
```

The simulation strategy is usually a mix of time slicing and event scheduling. Time slicing is used to routinely check each object for its responses to any change in the world state. Event scheduling is needed to allow objects to schedule any delayed response or action that is to occur at some future time which may not necessarily coincide with a particular time slice. The simulation proceeds until the termination criteria—such as goal success or failure—is met. The necessary output data of the simulation is now sent to the Evaluator block.

2.3 Experimental Design Block : Executive

In simulation, experimental design is a method of choosing which configurations (i.e., parameter values)

to simulate so that the desired information can be acquired with the least amount of simulation (**?**). The number of runs, S, is a function of the total number of factors, the number of factor levels and the number of replications (repetitions). For a typical route traversal simulation we can express the computational complexity of a single run for route R_j, in the worst case, as follows (given that we are using time slicing), $\phi(R_j) = O\left((Num_{so} + L)\frac{dist_{R_j}}{\Delta dist}\right)$ where $dist_{R_j}$ is the total length of route R_j; $\Delta dist$ is the size of time slice $\Delta t * Speed(t)$; Num_{so} represents the number of stationary objects previously calculated; and L denotes the total number of moving objects in the simulation. Thus, the total time complexity of the experimental part of the SBP algorithm will be, in the worst case, $O\left(\sum_{j=1}^{N} S \cdot \phi(R_j)\right)$.

In summary, we can save time in two ways: either by reducing S or by reducing $\phi(R_j)$, while still obtaining meaningful results. Heuristics in reducing S include: reducing the number of factors, reducing the levels of an ith factor and reducing the number of replications. For our problem domain, routes, behaviors and the planner's speed are candidate factors. By setting any one of these factors to be a constant, we are effectively reducing the number of factors. For instance, using some heuristics, we can prune away unpromising routes before they are simulated and thus reduce the number of levels of the route factor. The number of replications depends mostly on the type of output analysis method used. Some different output analysis methods we have employed are discussed in Section 2.4. The number of replications can also be reduced by *heuristic* sampling or *controlled* sampling of the uncertainties. This way, we can converge on the answer faster than sampling purely by random. The number of replications will partly depend on the randomness of the data. In effect, the number of uncertain objects in an alternative greatly affects the number of replications needed to reach an acceptable level of accuracy.

Next, we discuss some ways to reduce $\phi(R_j)$, the total simulation time spent during the evaluation process. In time critical situations, the quality of the desired information may be sacrificed to meet a given time constraint. Increasing the size of the time slice Δt (assuming the simulation is based on time slicing) is one way of decreasing the simulation time. Speed up will result at the expense of accuracy since the longer the time slice, the faster the simulation. Reducing (Num_{so}) to be simulated for route R_j can also improve the simulation speed. This can be achieved basically by calculating an *effective* set of objects–objects that are in the area of moving objects. More in depth discussion appears in (**?**). We can also save time by using aggregated or abstract models where possible. Currently, research is underway to allow simulation at levels of abstraction (**?**).

By distinguishing routes based on their distance, we can also build experimental strategies based on this information. Depending on the particular problem domain, the effect of the route distance on the simulation time will vary and thus, the Executive must make a decision as to where the time must be saved, either in S or $\phi(R_j)$s, depending on which is dominant.

We now present our approach which can produce simulation results within a given time constraint. The approach is best explained in two phases. In the first phase, we perform a set of n replications. It is hard to tell what is a good value for n but 20 is a commonly used number in simulation (**?**). While the n replications are being performed, the CPU times of each replication for each route R_j are recorded (this is measured by calling the system call clock() and calculating the elapsed time between the start and the end of the replication). An issue exists as to whether CPU time is a good measure since it can vary depending on the load of the computer or the network. In the second phase, we make a decision as to which output analysis approach (strategy of performing replications and choosing the appropriate stopping conditions) to use based on the expected simulation time and the current remaining time. The basic idea of the algorithm is to perform the needed number of replications–given that the remaining time is sufficient for their completion–for each route obtained from the SelectBestk algorithm which computes the exact number of replications necessary to reach a decision (select the best alternative among k alternatives) for alternative j. If the remaining time to perform the simulations is not sufficient, then we perform as many replications as possible given the time allowed. As the replications are performed iteratively, we also try to eliminate any alternatives that appear to be significantly worse than other alternatives. With the latter approach, the idea is to incrementally converge on the answer while trying to meet the time constraints.

2.4- Output Analysis Blocks :- Replicator, Evaluator, Analyzer

Specifying the right types of statistical analyses is just as important as performing the right types of simulation runs. With simulation, several different interpretations can be obtained from the same output data. Different analysis methods apply depending on whether a simulation is *terminating* or *steady state*. Because plans have a definite start and an end

time, ours are terminating simulations. Suppose we are simulating k alternatives (or routes in our case). We describe in the following our current strategy for obtaining the appropriate outputs and their analyses.

2.4.1- Replicator

Replication provides the easiest form of output analysis.- An issue is using *common random numbers* (CRN) to provide a controlled environment for comparison among alternatives. We use it here across different alternative route plans within the same replication to eliminate any "environmental differences" that can exist between different simulations. Depending on how the Trial block proceeds with the simulation, the designer may choose to vary the random number streams either in between each execution of the Trial block or within a single execution of the Trial block. In the air force route planning system in (**?**), for example, a single execution (or replication) of the Trial block consists of simulating all alternatives using a common random number seed. The next time the Trial block is invoked, the Replicator will provide a different random number seed so that a different environment will be created in the next replication.

2.4.2- Evaluator

In its simplest form, the Evaluator serves as the accumulator of any relevant simulation data that is produced from the Trial Block. If the objective function within the Trial block produces a set of scores for each alternative, a straightforward way is to total the scores produced from the replications for each alternatives. Other relevant data such as elapsed CPU time for simulation of each alternative may also be accumulated so that the Executive block may later analyze and predict future time usage.

2.4.3- Analyzer

Based on statistical criteria (e.g., highest mean, smallest variance), we can consider several alternate plans and choose the "best" plan for execution. Criteria other than statistical in nature can also be imposed, that are based on heuristics or expert knowledge. This block is primarily responsible for analyzing data that was accumulated in the Evaluator. For most of our applications, the averages of the replication results serve as the basic "data" points in our response surface or graph, representing the goodness of a plan. Using the confidence-interval approach, there are mainly two ways to compare among k alternatives that are discussed in the literature (**?**). Since we are trying to select the best out of k alternatives, we must detect and quantify any significant pairwise differences in their means. The first "iterative" approach, although quite accurate in terms of the results, can include an unnecessarily large number of replications due to the fact that a constant number of replications is performed uniformly across the current set of alternatives. The second approach, which we call the "non-iterative" approach (also called the "Selecting the Best of k Systems" approach) selects one of the k systems or alternatives as being the best one while controlling the probability that the selected system really *is* the best one. The approach is discussed in detail in (**?**).

3- NONDETERMINISTIC ADVERSARIAL ROUTE PLANNING EXAMPLE

As one of the applications of the SBP, we have chosen a typical air interdiction scenario, and developed its Simulation Based Planner (C++) and GUI (Tk/Tcl) under our Multimodeling Object-Oriented Simulation Environment (MOOSE). Interdiction mission is a typical air mission where the purpose is to destroy, delay, or disrupt existing enemy surface forces while they are far enough from friendly surface forces that detailed coordination of aerospace and surface activities is not needed.- However, difficulties arise because methods of penetration can vary according to the strength and sophistication of the enemy's detection, reporting, command and control network, and how much intelligence is available about its capabilities. Considering these uncertainties and constraints, selecting the best route is a very difficult task.

Figure 2 defines a scenario with dynamically moving objects. The mission of the blue (friendly) force aircraft is to destroy a red (enemy) force munitions factory. There are three Radars (*R1, R2, R3*) and two Surface-to-Air Missile (SAM) sites (*S1, S2*), each with different effective detection ranges.- Two red force aircrafts (*A1, A2*) are located in air defense zones Zone2 and Zone3 respectively, while one red force aircraft (*A3*) is located outside of the air defense zones. At a first glance, the problem of guiding the blue force around the radar, SAM and air defense zone coverages, and toward the factory seems like a simple problem in computational geometry. In fact, this is the manner in which most route planning is done. A typical rule might be formed "To locate a path, avoid radar and SAM fields, and avoid fighting against enemy fighters." However such a rule based reasoning becomes more onerous when uncertainty and dynamics are present. For each simulation trial, the uncertainty of *S2* is handled by first sampling a random location for *S2* within the boundaries of the circle drawn around *S2*. Taking this location

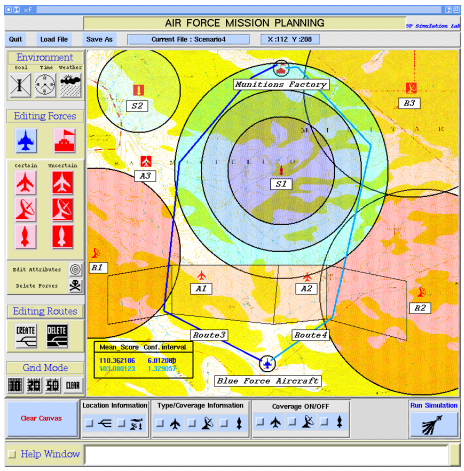

Figure 2: A Typical Air Interdiction Scenario

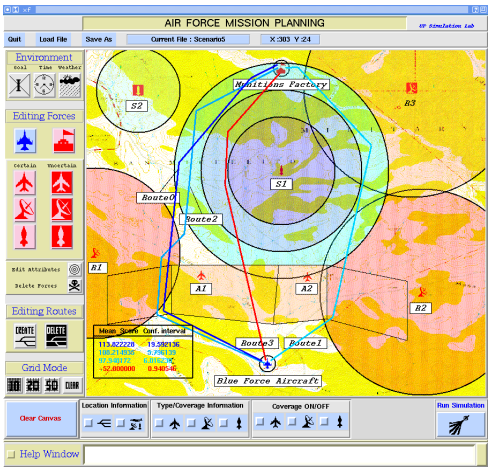

Figure 3: Inserting Two New Routes

as the center point of the SAM site, a boundary circle is drawn representing the arm range of the SAM site. The uncertainty of the radar *R3* is handled in a similar manner. The location is first determined by sampling the point within the uncertainty circle drawn by the user. Using the sampled point as the center point of the radar, a boundary circle is drawn representing the detection range. For the objects in our air force mission planning domain, we have concentrated mainly on location and range uncertainties, even though other types such as uncertainty of mission, fire power and existence can also exist.

4 RESULTS

Figure 2 shows two possible routes (*Route3, Route4*). The goal of blue force aircraft is to destroy the red force munitions factory while satisfying three constraints: time or fuel level, safety, and destruction of the target. Given the possible routes, the role of the simulation-based planner is to choose the best route minimizing time and fuel consumption, and maximizing safety and target destruction. *Route3* was chosen to avoid direct attack from *A1*, but for a short time period it will be detected by *R1*. *Route3* also takes the blue aircraft into the track range of *S1*, but not into its arm or missile range. Being detected in the track range of *S1* is not very dangerous since only tracking functions may be performed by *S1*. Overall, we expect the success rate of route 3 to depend largely on the result of the samplings for uncertainty factors: specifically, the location and guidance capability of SAM *S2* and the mission type of *A3*. If the powerful guided system of SAM is sampled close to this route, or *A3* has an intercept capability, then the chance of success will be very small. Otherwise, the chance of mission success will be very good. These nondeterministic and stochastic characteristics are resolved by multiple simulations using different samplings of the uncertainty factors. *Route4* was carefully chosen to minimize the amount of time that a blue force aircraft falls within the detection ranges of *R2* and *R3*. The result of the SBP shows almost the same mean score for *Route3* and *Route4* (*Route3* : 110.36, *Route4* : 103.08) with *Route3* being slightly better (the goal is to maximize the mean score for determining the better plan). Intuitively, *Route 4* seems like a better route since it only involves radar sites whereas *Route 3* has a SAM site *S2*, although its location may be uncertain. With simulation-based planning, however, we discover that *Route 3* is slightly better. But depending on our objective, we may select *Route4* as the best overall route based on its narrower confidence interval (*Route4* : 1.3, *Route3* : 6.0). Next, in order to reduce the total number of replications in the simulation, we compare two different methods of output analysis. The first one, which we refer to as the "iterative" method, attempts to quantify significant pairwise differences among the k means within a given confidence interval α. We call it "iterative" because of the fact that the algorithm iterates–performing for every iteration, a set number of b replications and analyzing data to see if there are any significant differences. Whenever a route is found who is significantly worse than all the other routes, it is eliminated. The iteration continues until only two routes remain and a difference exists between the two of them. Note that since we have 4 routes in our experiment, each confidence level for the pairwise differences must be made at $1 - \alpha/6$. When two alternatives are actually very close together, in terms of their goodness, we might not care if we erroneously choose one system (the one that may be slightly worse) over another (the one that is slightly better). Thus, given a "correct selec-

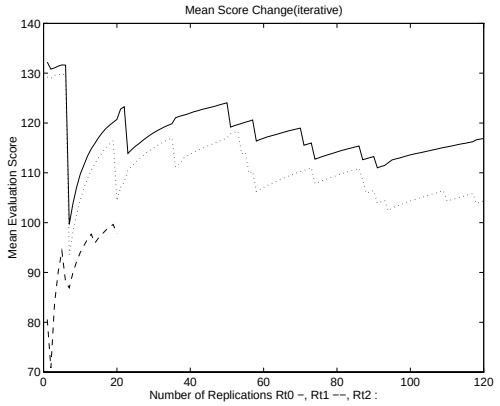

Figure 4: Mean Score Changes of 3 Routes: Iterative Method

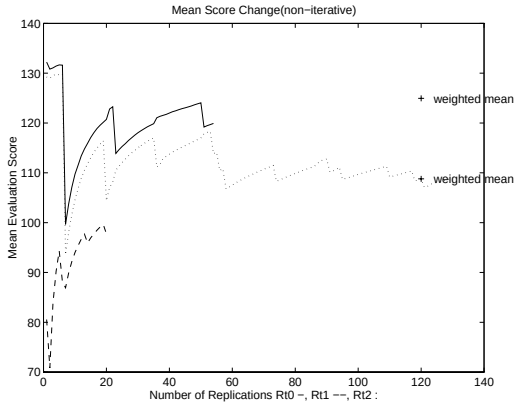

Figure 5: Mean Score Changes of 3 Routes: Non-Iterative Method

tion" probability P^* and the "indifference" amount d^*, the "non-iterative" method calculates how many more replications are necessary in order to make a selection–a selection where with the probability at least P^*, the expected score of the selected alternative will be no smaller than by a margin of d^*. In the following experiment, we have chosen $P^* = 0.95$ and $d^* = 13$. A smaller d^* will produce more accurate results but with many more replications.

In addition to the two routes that appear in Figure 2, we add two more routes to test how much the number of replications reduce and also if the identical selection is made. The routes are shown in Figure 3 and are renumbered. *Route 2* and *Route 0* represent two alternatives that are very close together so that it will likely require many replications to quantify a significant difference between them. As expected, Routes 0 and 2 do exhibit similar responses as shown in the following plots. Figures 4 and 5 are results with just 3 routes: 0, 1 and 2. Figure 4 shows the mean score change of the routes using the simple iterative method. After the first 20 replications, the planner decides that route 1 can be eliminated since its scores are in general significantly lower than those for the other two routes. It then performs 120 replications for both routes 0 and 2 before it decides that there is a significant pairwise difference in their means to make a selection. Using this method, we perform in total 60 replications. With the non-iterative method, although the method decides that 10 more (130) replications are needed to make a decision on route 2, a smaller number of replications is made in terms of the total number–$130 + 54 + 20 = 204$. In this particular scenario, both the iterative and the non-iterative methods select route 0 as the best alternative.

Now, we add a 4th route, *Route 3* which is the shortest but perhaps the most dangerous route. Figures 6 and 7 show the mean score changes for these 4

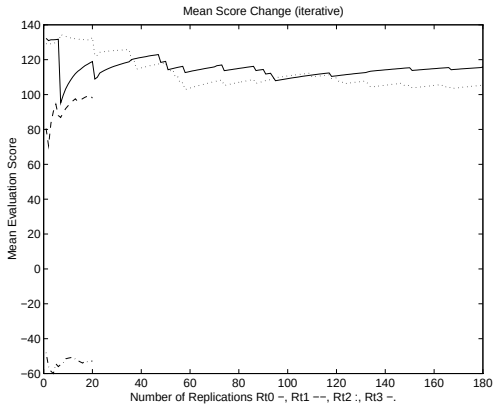

Figure 6: Mean Score Changes of 4 Routes: Iterative Method

routes. Route 1 and 3 are eliminated after 20 replications since the average scores are significantly lower than routes 0 and 2. With another route added, the mean score change plots are somewhat different than in the case where there were only 3 routes. And this difference pushes the iterative method to continue replicating 60 more times for each of the two routes before it makes a decision. Consequently, it chooses route 0 as the better route–a different selection than when only 120 replications were performed. In the non-iterative method, it makes only 69 replications for route 0 and 21 replications for route 2 before it makes a selection. The method chooses route 2 to be the best route in this particular case. As discussed in (**?**), this can occur because the non-iterative method only ensures that it makes a correct selection within a given probability P^*. Overall, the iterative method performed 400 replications whereas the non-iterative method only did 130 replications.

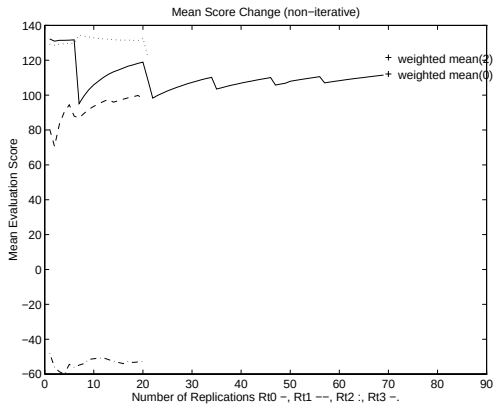

Figure 7: Mean Score Changes of 4 Routes: Non-Iterative Method

5 CONCLUSIONS

We have presented the method of simulation-based planning as a new approach to route planning under uncertain and complex environments. In particular, for the area of multi-agent planning, SBP can break down the complexity of reasoning by distributing the reasoning tasks to individual objects using object-oriented multimodel simulation. SBP goes about its business in a way that is considerably different from the *cognitive model* method that attempts to model each individual agent in terms of a knowledge-base and reasoning engine. The behaviors of humans in tightly constrained environments or in emergency conditions has successfully been modeled without employing cognitive modeling. To be flexible and realistic, we acknowledge that the hypothesis of the *ineffective cognitive model in a crowd/multi-agent scenario* may not always hold true. We are unsure of how effective SBP would be in scenarios which are not as tightly controlled and orchestrated as command and control in the military. For these reasons, it is prudent to view SBP as one method of handling decision making for multiple agents. Possibly, a hybrid SBP/cognitive modeling approach may yield the best decision-making results.

6- FUTURE WORK

We must address the issue of the validation of simulation models–making sure that the models we build appropriately represent the actual object's behavior. A common approach is using sensitivity analysis or an inspection by an expert. More detailed, sophisticated models should be built to obtain better results in terms of answer quality and also test the degree of CPU time consumption in respect to the model's complexity. An immediate future work would be to extend the implementation of the Air Force models to include all the levels of abstraction. Larger numbers of objects should also be simulated to further study the scalability and the rate change of time consumption. Extending the multimodeling paradigm to enable model execution at any level of abstraction is also currently underway and SBP can greatly benefit from the success of this work since it will allow reduction of model execution time. Possibilities exist for future work in finding other ways of meeting real-time constraints: a hybrid approach of using quantitative and qualitative (fuzzy) simulation, developing additional heuristics to aid in optimizing the simulation process are some ideas that we plan to research in the future. Finally, to further extend the study of the SBP methodology, additional experiments in other application areas should be performed. Also, building and comparing two planning systems, one built using the SBP approach and one built using another planning approach, should prove to be useful in further improving the SBP approach.

ACKNOWLEDGMENTS

We would like to thank the following funding sources that have contributed towards our study of modeling and implementation of a multimodeling simulation environment for analysis and planning: (1) Rome Laboratory, Griffiss Air Force Base, New York under contract F30602-95-C-0267 and grant F30602-95-1-0031; (2) Department of the Interior under grant 14-45-0009-1544-154 and the (3) National Science Foundation Engineering Research Center (ERC) in Particle Science and Technology at the University of Florida (with Industrial Partners of the ERC) under grant EEC-94-02989.

AUTHOR BIOGRAPHIES

JIN JOO LEE is a research scientist in the Interconnect Modeling group at LSI Logic Corp. She also holds an Assistant Professor position at San Jose State University. In 1988, she received a B.S. degree in Computer Science from Ewha Womans University, Korea and in 1991, an M.S. degree in Computer Science from Brown University. After receiving the M.S. degree, she was a research engineer at Human Computers Inc., Korea until 1992. She received a Ph.D. degree from the Computer and Information Science and Engineering department at the University of Florida in 1996. Her research interests are in planning in Artificial Intelligence, simulation and control.

PAUL A. FISHWICK is an Associate Professor

in the Department of Computer and Information Science and Engineering at the University of Florida. He received a BS in Mathematics from the Pennsylvania State University, MS in Applied Science from the College of William and Mary, and Ph.D. in Computer and Information Science from the University of Pennsylvania in 1986. He also has six years of industrial/government production and research experience working at Newport News Shipbuilding and Dry Dock Co. (doing CAD/CAM parts definition research) and at NASA Langley Research Center (studying engineering data base models for structural engineering). His research interests are in computer simulation modeling and analysis methods for complex systems. He is a senior member of the IEEE and the Society for Computer Simulation. He is also a member of the IEEE Society for Systems, Man and Cybernetics, ACM and AAAI. Dr. Fishwick founded the `comp.simulation` Internet news group (Simulation Digest) in 1987, which now serves over 15,000 subscribers. He has chaired workshops and conferences in the area of computer simulation, and will serve as General Chair of the 2000 Winter Simulation Conference. He was chairman of the IEEE Computer Society technical committee on simulation (TCSIM) for two years (1988-1990) and he is on the editorial boards of several journals including the ACM Transactions on Modeling and Computer Simulation, IEEE Transactions on Systems, Man and Cybernetics, The Transactions of the Society for Computer Simulation, International Journal of Computer Simulation, and the Journal of Systems Engineering.