Simulation-Based Planning for Computer Generated Forces

Jin Joo Lee , Paul A. Fishwick
Department of Computer and Information Sciences
University of Florida

Abstract

Planning in AI has been an active research topic for more than thirty years but only recently has it started to move in the direction of combining planning and execution to achieve what is sometimes called as 'Intelligent Reactive Planning'. We propose simulation-based planning as a new way to perform intelligent reactive planning. Simulation-based planning —unlike most other planning systems—integrates simulation into the planning process. Once a set of plans is generated, simulations are used to test and evaluate the plans to choose the most applicable plan for that current situation. In most planning systems, plan evaluation depends on rules alone and because rules must be designed general enough to cover all possible cases, the evaluation is not specific enough for some individual cases. However, when the plan evaluation is done through simulations, the evaluation can be more fine-tuned to individual cases and can allow better plans to be chosen for that individual case. From the military planning perspective, the simulation-based planner is also quite useful due to its ability to perform adversarial and multiagent planning. This is a natural consequence of using simulation in the planning process. By allowing other entities such as the enemy to simulate in parallel with the planner's forces, the planner is able to observe, prior to the actual execution, the effects of adversarial and multiagent actions against its own plans.

1 Introduction

Planning in Artificial Intelligence has been a long standing problem where researchers have tried to build automatic systems that describe a set of actions which, when executed, will lead to a desired goal. Many of the early planners had a different view of planning than how we view planning recently. They viewed planning mainly as a separate entity from execution and assumed there is only one active agent in the world at any time. Some of these earlier classical planners are STRIPS [5, 6], NOAH [13] and SIPE [15].

These classical planning approaches will work well when one can assume that the planner has complete knowledge of the current state and the cause-and-effect relationships of every action in that world. However, for situations which cannot guarantee the above 'ideal' condition—which is the case for any real world planning problems— the common planning approach will not be sufficient. The actual execution of a generated plan is no longer guaranteed to succeed. This can occur whenever there are changes that can take place in the environment over which the planner has no control, such as another agent or an enemy, and when there is uncertainty of available information or uncertainty of another agents reaction. Accurate prediction of the resulting states of plan execution will be difficult. Once concepts such as multiagent and adversary are introduced into planning, the classical approach will either break down in the plan generation process, or produce unrealistic plans that can only succeed in an ideal (artificial) environment. As more variables are introduced, a large increase in complexity of reasoning is unavoidable, especially when one central system is responsible for reasoning about all possible changes in the environment. To overcome this increase in complexity of reasoning, many new approaches have been introduced [14, 3, 10, 9]. Apart from the differences in methodology, some properties exist that are common to many planning systems which have been built for real applications such as [1]. One property is that the planning process divided into two major phases. During the first phase, a set of candidate plans that satisfy some subset of constraints is generated (called *plan generation*). Then in the following phase, a solution plan that satisfies the remaining constraints to its best is chosen. A strategy of dividing the constraints may be to divide them according to their criticality to the success of the plan. Another property is that some form of Temporal projection is used in each phase. *Temporal projection* is projecting into the future, to find out if a sequence of actions specified in a plan is applicable in a particular situation and if so, to predict what the results will be after the execution. In the first phase, temporal projection is used to find a set of applicable plans. Then in the second phase, it is used to evaluate the set of candidate plans by using an evaluating function which measures different properties of the plan such as probability of success and loss of resources. Users can obtain different results by giving different evaluation criterion each time the problem is given to the planner. In general, one or more rule-based systems are used to generate and evaluate the plans at each phase. Our approach differs in the second phase. Instead of performing the temporal projection of plans using rules, we perform a simulation of each plan using simulation models.

We propose simulation-based planning as a new concept through a sample application in the domain of mission planning for Computer Generated Forces(CGF). Our mission planner is an integral part of a larger project of the Institute for Simulation and Training (IST) called "Intelligent Autonomous Behavior by Semi-Automated Forces in Distributed Interactive Simulation" which is funded by the U.S. Army Simulation Training and Instrumentation Command (STRICOM). The goal of the planner is to automatically derive plans for a semi-automated force (CGF), at the company level initially, so that the force will provide an Army trainee with an effective training experience. Planning is only a small part of the overall project, which includes efficient line of site (LOS) determination, terrain reasoning, intelligent target acquisition and behavior representation for CGF entities. The planner takes orders from the battalion level and translates these orders, with a tight coupling with the terrain analyzer, into efficient plans for the CGF platoon entities. In addition to planning for its subordinate units, the planner must also also be able to monitor the execution of the plan, react to unexpected situations and replan if necessary.

Section 2 discusses simulation-based planning as a general concept. We describe the architecture and design of our planner in section 3 using a sample mission to help illustrate our methods. Future work is discussed in section 4 and finally some conclusions in section 5.

2 Simulation-Based Planning

The simulation-based approach to planning has been used by the military for a considerable time in the form of conflict simulation (or wargames). Conflict simulations are termed *constructive simulations* when comparing them against the *virtual* or *live* methods. Simulations of the constructive type involve aggregate simulations using discrete time (turns) and space (terrain). During a planning phase, a commander would perform "what if" scenarios by setting up a course of action on a hexagonally tiled map of the terrain. Engagements, instead of being fought with individual entities, are abstracted using a stochastic method in the form of a combat results table (CRT) or through Lanchester equations for force attrition.

The wargame approach to planning is simulation-based, where the model is aggregate and the simulation is discrete and "turn" oriented (in a game fashion). Related work by Czigler et al. [2] demonstrates the usefulness of simulation as a decision tool. Our approach is to extend this method by using more detailed models, where the models simulate entity queuing at fords and bridges as well as engagements. At first, our engagement simulation will use many of the features of constructive models, such as probabilistic combat results tables, but we will eventually create plans that involve simulated entity-level interaction since this is the most accurate way to learn if a plan will fail or succeed.

In the simulation literature, simulation is defined as " the discipline of designing a model of an actual or theoretical physical system, executing the model on a digital computer, and analyzing the execution output"; Fishwick [8]. In Dean [4], it states that the idea of using model to formulate sequences of actions is central to planning and given a sequence of actions, a robot can use the model to simulate the future as it would occur if the actions were carried out. So simulation provides the robot with information which can be used to suggest modifications or to compare the proposed sequence with alternative sequence. And therefore, once simulation models have been built for a system, simulation can be used as a tool to provide the system with information useful for evaluating its hypothesis. It is logical that we employ simulation within the planning process to gather information about each candidate plan (sequence of actions) and to compare them.

3 Overview of Planner

The biggest challenge for Computer Generated Forces (CGF) systems is to simulate human behavior. Our particular interest is to simulate the behavior of a company commander which involves mission planning and plan execution for its company. Given an Operation

Order from a higher level unit, the planner's role is to generate a mission plan that will achieve the task(s) given in the Operation Order. Using this plan the planner then issues the corresponding subtasks to each subordinate unit, which in this case are platoons, in the form of an Operation order.

3.1 Planner's interface to IST CGF Testbed

As mentioned earlier in section 1, the mission planner is a part of a larger project at IST. The planner must integrate with many other components in the IST project, but for the most part it must work with the IST's Terrain Analyzer.

3.1.1 Terrain Analyzer

The Terrain Analyzer is the planner's only source of information where terrain is concerned and thus the planner uses the TA quite extensively during the planning process. The TA is responsible for route planning, finding tactical positions, computing Line of Sight and answering questions about terrain features. The interface between the TA and the planner is established by four types of calls; ROUTE, DEF_TACT_POS, LOS and TERRAIN_FEATURE.

- ROUTE: Given the maximum number of routes, start and end position, the unit boundary, the minimum percentage of concealment, the mission type and the direction of approach to the OBJ (located at end position), the TA returns multiple routes that satisfy the given constraints. Note that the direction of approach does not apply to cases when the end position does not have an enemy unit and the mission type is not a SEIZE mission. Any intermediate enemy position that needs to be avoided can also be given along with a radius and the TA will generate routes avoiding these locations radius distance away from the avoid locations. Each returned route has a route id, length and percentage of concealment relative to the route. The actual route is represented as a piecewise linear curve made up of a set of line segments. Each line segment contains not only it's begin and end points but also the percentage of mobility, passable width, probability of LOS to the OBJ and the terrain_type (ground, road, ford).

- DEF_TACT_POS: Given the type of mission, this call requires the TA to provide locations for a given type of tactical position(s). for the type of tactical position, given the current type of mission. The current position of the unit and the OBJ must also be given There are three different types of position: SUPPORT_BY_FIRE, DEFENSE, and ATTACK. We have two types of mission : SEIZE and DEFEND. Finally, the enemy location must also be provided to the TA for SUPPORT_BY_FIRE and ATTACK positions.

- ALOS: This call directs the TA to perform an area to area Line_of_sight determination. Locations 1 and 2 are given, each with Radius1 and Radius2. The Radius1 describes the circular area centered at 1 and Radius2 describes the circular area centered at 2. 1_#_PTS constrains the number of points where LOS should be tested within circle centered at 1. 2_#_PTS constrains circle at location 2 in the same manner. If Radius is 0 for both locations then a simple point to point LOS is performed. The returned data is the probability of sightings over the #_PTS tested within the two areas.

- TERRAIN_FEATURE: This call requires the TA to return all the terrain features that are included within the radius of a given point. The planner supplies the TA with a center point and a radius and the TA returns one or more of the following types as a terrain_feature: ROAD, FORD, GROUND, CHOKE_POINT.

3.2 Planner Architecture

Figure 1 displays the architecture of our planner in relation to the IST CGF Testbed [11]. Each commander in the IST testbed is simulated by a Command Entity (CE). The Planner performs major functions of this entity. The planner has two phases: the Reactive phase and the Planning phase. A Phase is a group of states that collectively display a behavior. Only one phase is active at any given time. The starting phase is the Reactive Behavior. The decision as to which phase becomes active is made by the current active phase based on the inputs. There is no single 'main' algorithm that controls the whole process. Thus, the decision is made in a distributive manner. In particular, the decision to give up planning and report to higher level unit is made by the planning behavior. The inputs are either Oporders or Sitreps, and depending on the type of Sitreps and Oporders, different decisions will result. Each Planner in the CE is made up of the following components.

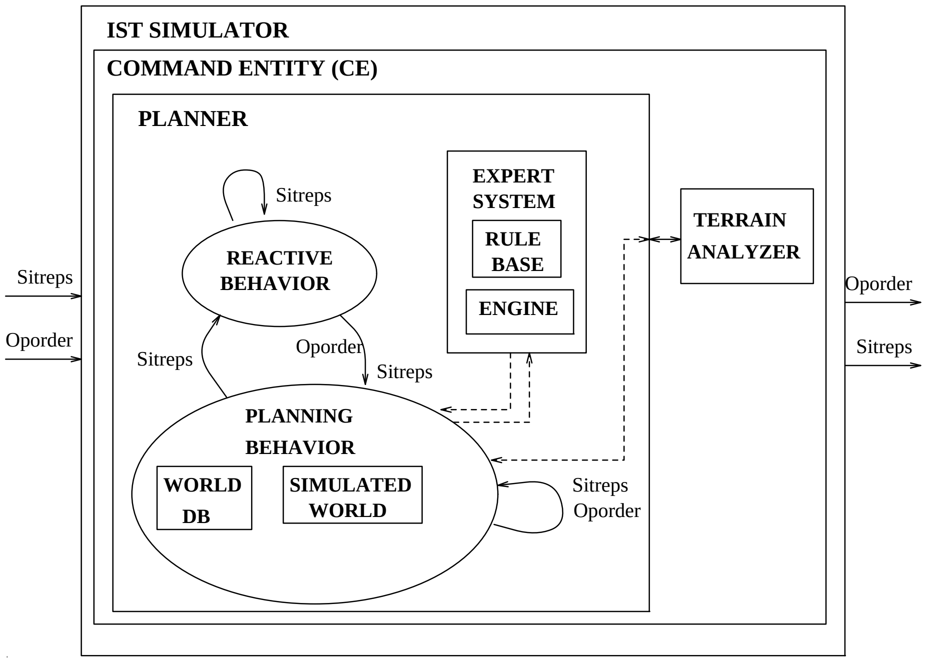

Figure 1: Planner Architecture

3.2.1 World Database(DB)

The World Database contains information about the battlefield. This is not a complete spatial representation of the battlefield (since the TA has this information) but a simplified database which mainly contains information that is known to the CE (and not known to the TA). Since the TA does not have any information regarding the location of enemy or friendly units and will not keep track of the locations, the planner needs to keep track of the locations and the status of the units in the World Database. This database is created as soon as the CE starts to exist. Initially it contains its own location and will be updated with new information as they become available to the CE via Sitreps or Oporders.

The battlefield is divided into rectangular regions and represented by elements of a matrix. This is a very low resolution of the battlefield because the purpose of this matrix is mainly to speed up the look up time of unit locations and other information by organizing the linked lists into regions. Each element of the matrix is linked to a linked list that contains all the information the CE has about that region. Each node will have more exact locations along with other available information such as status of the unit.

3.2.2 Reactive Behavior

The Reactive Behavior module displays reactive behavior necessary for survival when immediate action is required. This behavior may be different for different types of entities. The module is initialized with a generic set of behaviors at the start and may be modified with any reactive behaviors provided by an Oporder.

3.2.3 Planning Behavior

The Planning Behavior module has the ability to generate orders for its subordinate entities from an Oporder given by a higher level entity. This module is made up of the following smaller modules where the order in which they are presented actually coincides with the algorithm steps of a typical planning process. For each step we will give a general description and illustrate via a sample mission. The mission in the sample Oporder given to the CE will be "SEIZE OBJ at (32.5, 28.5). Friendly units at Assembly Area(AA) located at (50.0, 52.5). Company UNIT_BOUNDARY is ((57.5, 22.5),(57.5, 57.5),(20, 22.5), (20, 57.5))." Figure 2 shows the terrain and the locations of the enemy at OBJ and friendly forces at AA. There is a river that runs through the middle of the terrain where there are three crossable fords. There is a lake near the enemy location. There are also some treelines and canopies in the terrain.

1. **Sitrep/Oporder Analyzer (SOA)** parses the Sitrep/Oporder to update the World DB.

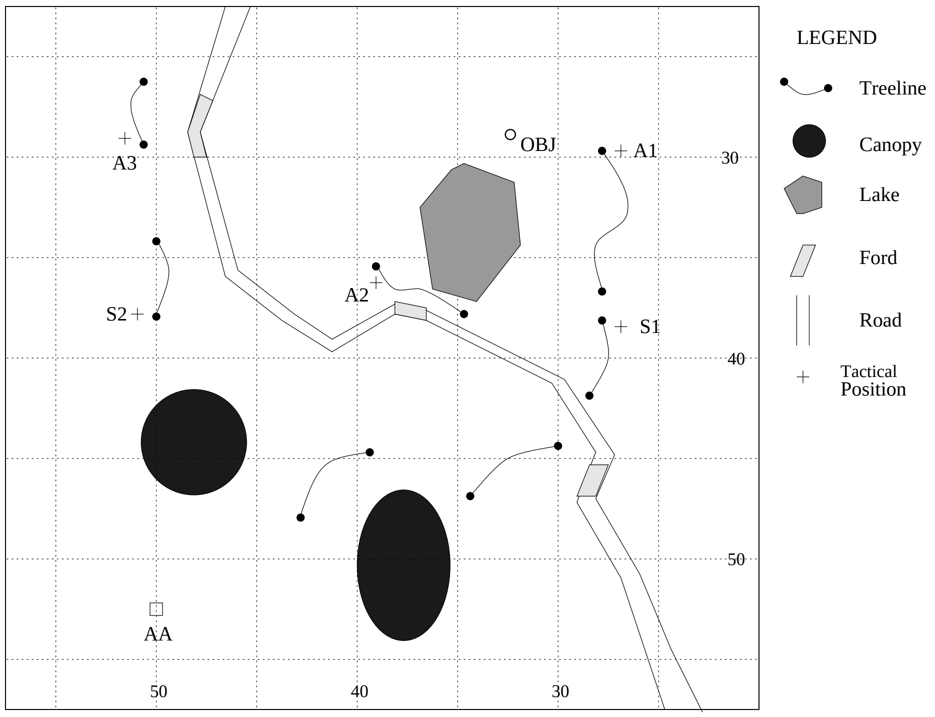

Figure 2: Sample battlefield

- The World DB will now contain locations of the enemy and friendly forces.

- **Oporder**: is further parsed to generate a list of task(s) to be achieved. The Situation Analyzer is called next with this list. The generated list of subtasks to be achieved for SEIZE mission will consist of the following list where the first list of subtasks is necessary to achieve the goal whereas the second is optional.

  1. MOVE to (ATTACK position) then ATTACK (OBJ)
  2. MOVE to (SUPPORT BY FIRE position) then FIRE (optional).

- **Sitrep**: Sitrep is analyzed to decide if any immediate action is required, if any replanning is required, or if any Sitrep needs to be generated. The Execution Monitor is called with the decision. We will omit Sitreps in this example.

2. **Situation Analyzer(SA)** is a collection of rules that analyzes the given situation using the World DB. Appropriate constraints for the ROUTE or the DEF_TACT_POS call are generated. The decision as to which of the two calls to call initially depends on problem size reduction. In other words, in a given situation, the call to ROUTE may produce many possible routes whereas a call to DEF_TACT_POS may produce only a few number of tactical positions. In this case, we can first call DEF_TACT_POS to acquire a set of tactical positions and then call ROUTE with these tactical positions which reduces the number of returned routes due to the given constraints. Thus, the SA performs some alternate calls of ROUTE and DEF_TACT_POS to produce an appropriate number of alternate routes. The Course of Action(COA) Tree Generator is called with these alternate routes. Following are the set of calls that are made for our sample mission. Figure 2 shows the the tactical positions as '+'.

   - DEF_TACT_POS:
     Type of position - ATTACK,
     Type of mission - SEIZE,
     OPFOR position - 32.5, 28.5
     TA returns position A1, A2 and A3 as ATTACK positions.

   - DEF_TACT_POS:
     Type of position - SUPPORT_BY_FIRE,
     Type of mission - SEIZE,
     OPFOR position - 32.5, 28.5
     TA returns position S1 and S2 as SUPPORT_BY_FIRE positions.

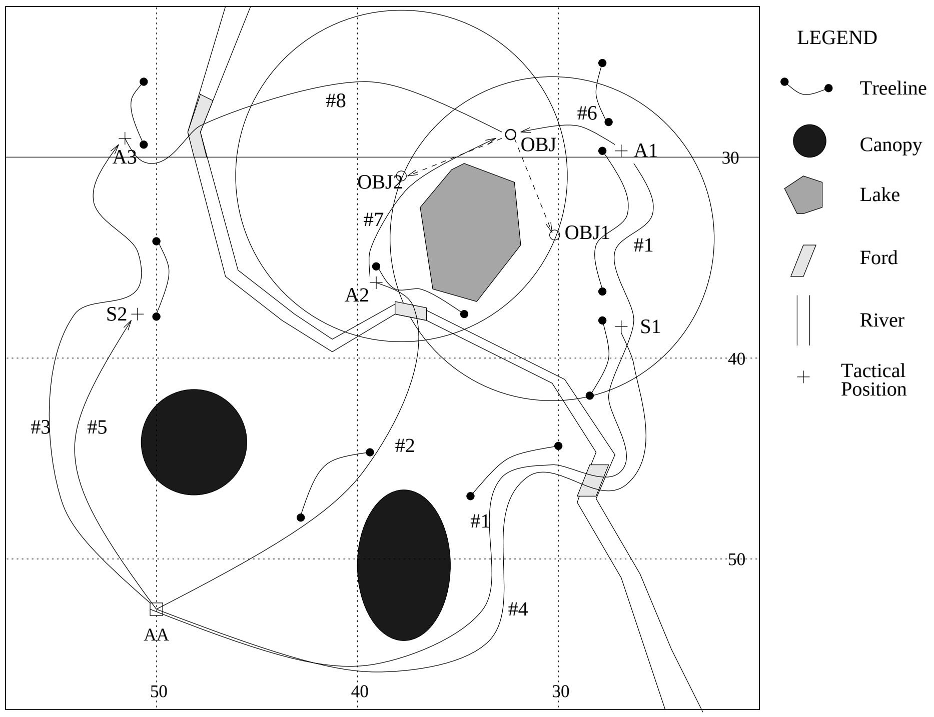

Figure 3: Sample battlefield

Using the generated tactical positions, the planner makes ROUTE calls to the TA as follows. Figure 3 shows the routes that are generated from these calls.

- ROUTE:
  Start, end position - AA, A1,
  Unit boundary - ((57.5,22.5), (57.5,57.5), (20,22.5), (20,57.5)),
  Type of mission - MOVE
  Any parameters that are missing are assumed to be unspecified by the TA. TA returns Route #1.(Note that more than one route may be returned by the TA for such a request).
- ROUTE:
  Start, end position - AA, A2,
  Unit boundary - ((57.5,22.5), (57.5,57.5), (20,22.5), (20,57.5)),
  Type of mission - MOVE
  Route #2 is returned.
- ROUTE(AA, A3, unit boundary, MOVE). Route #3 is returned.
- ROUTE(AA, S1, unit boundary, MOVE). Route #4 is returned.
- ROUTE(AA, S2, unit boundary, MOVE). Route #5 is returned.
- ROUTE(A1, OBJ, unit boundary, ATTACK). Route #6 is returned.
- ROUTE(A2, OBJ, unit boundary, ATTACK). Route #7 is returned.
- ROUTE(A3, OBJ, unit boundary, ATTACK). Route #8 is returned.

3. **COA Tree Generator** uses the set of alternate routes produced by the SA and generates a COA Tree. The COA tree is a tree of alternatives. It is a tree that contains every possible combination of alternative choice or action of the company. Figure 4 shows the COA tree built from our example. It is partially drawn since many of the branches are redundant. The 1st level of the tree allows the choice of splitting or not splitting the company in achieving the mission. If the company is to be divided, it means that one or more platoons are to support the attacking unit. For this example, the supporting role is SUPPORT_BY_FIRE. The second level contains the alternate routes available for the units. Note that the available alternate routes for the case of not splitting the company, the routes that lead to SUPPORT_BY_FIRE positions (#4,#5) is not considered. The next level contains alternatives for different formation such as Tactical Road March and Bounding Overwatch. The next

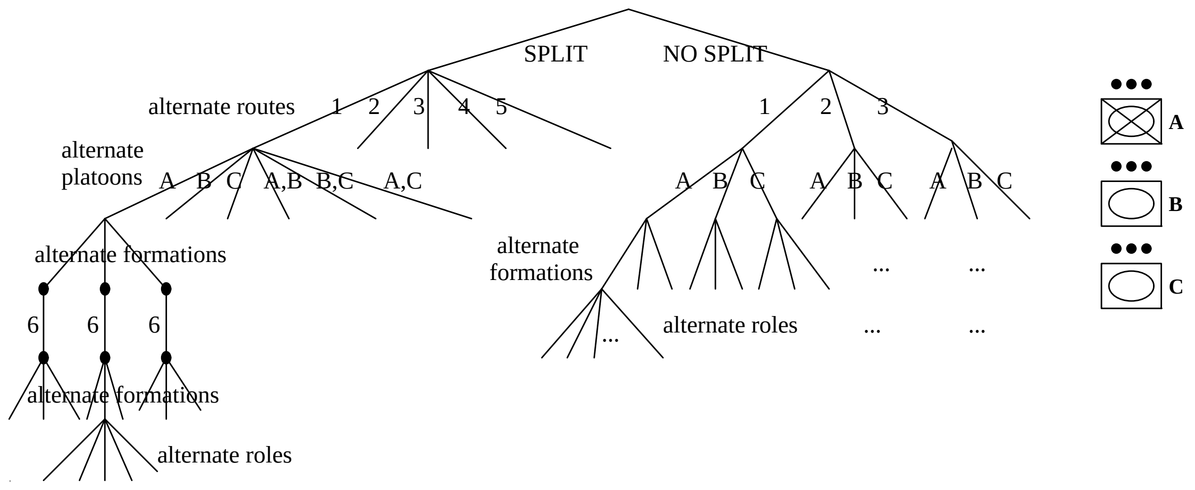

Figure 4: Sample COA tree

level contains other alternatives such as varying the role of platoons in different formations. Also note that after the choices are made for route #1, the next level starts by choosing the next connecting route (from #1 to the OBJ which is #6 in this case). Similarly for #2 and#3, #7 and #8 will be the connecting routes respectively. We can extend the tree as much as we want with any other possible alternatives. Some alternatives can be omitted at this stage and later generated during the Simulation step.

After such a tree is generated, the tree is pruned using various methods and rules before it is passed onto the COA Simulator. Many alternatives can be pruned away by using a military expert knowledge system. However, we must not prune away too much since many alternatives should be left to be explored via simulation. Otherwise the purpose of using simulation may be lost since most of the choice would have been made already. Next, the COA Tree Simulator is called with the COA Tree.

4. **The COA Tree Simulator** is invoked to simulate the set of COA trees that have been generated. This is done by creating a Simulated World and performing the simulation of friendly and enemy units by time slicing between actions (move, look, fire) and observation by each unit. It is also time sliced between friendly and enemy units. Different methods can be used in simulating friendly and enemy units. One method is to allow the enemy units to have the same planning capabilities as the friendly units but with different tactics. This method would be quite realistic, but it can be quite time consuming. In general, a complete simulation is more time consuming than a temporal projection using rules. If computing capabilities are limited, we can perform simulation at different levels of abstraction [12, 7] where each higher level will use less computational power. Another solution is to let the enemy units follow a less sophisticated planning process allowing limited intelligence. This is the approach we are taking for this particular application due to limited computing capabilities. In our example (figure 3), the enemy unit at OBJ is simulated to react to the friendly units by either moving to location OBJ1 or OBJ2. The circle centered at OBJ1 represent the area of firing range of enemy when enemy is at OBJ1. It is the same case for OBJ2. Thus, when the two opposite forces are inside the circle, engagement is likely when the enemy has line of sight to our unit. The actual simulation algorithm is as follows:

```
While (planner active) do
    Update entity state variables
    Perform line of sight (LOS) check
    Engagement check
    Update current clock time by $\Delta T$
End While
```

For each course of action, the simulator operates as follows. State variables defining an entity's position and orientation are updated at each time slice. In low mobility areas or areas with a steep terrain gradient, the movement will change over time. Also, for some terrain features, as with fords or chokepoints, a simple queuing model can be executed to keep track of entities that must wait for entities that are blocking the path. Service times and speed values are obtained by sampling from a probability distribution appropriate for the blocked area. A line of site (LOS) check and range calculation is done between the entity being simulated and known enemy locations. If the enemy is within range of certain weapons (such as a HEAT or Sabot round), an engagement will ensue. The circles around OBJ1 and OBJ2 represent this range for each location, given that the enemy is predicted to have moved to either OBJ1 and/or OBJ2. We are unsure as to the level of detail required to simulate the engagement for planning purposes. Initially, our method will be to use attrition or combat result tables (CRTs); however, these will be extended as behaviors such as "seeking cover" are integrated into the planner. The simulation proceeds, while updating the simulation time by ΔT until either the plan has been fully simulated, or the planner is interrupted.

There are several advantages to using Simulation to predict the results of plan execution.

(a) Simulation provides a uniform method without resorting to adhoc solutions. In simulation, each entity in the environment is simulated in a uniform and consistent manner by using models that represent both the physical and behavioral properties. Thus, simulating a plan is a natural consequence of simulating each of the entities by itself without having to worry about the global state change as a result of each entities action. In some ways, simulation can be viewed corresponding to object-oriented programming methods. Thus, each object is simulated using its own model.

(b) Because there is no central reasoning node for the simulation but many individual simulation models for different entities, scalability is a natural consequence. Extendibility is another advantage simulation provides. For example, the effects of adding a new type of entity will be clear, only the behavior models of each entity must be updated to recognize and reason about this new entity.

(c) Similar to how simulation is used for visualization, simulation can be easily used to perform visual playback of how a plan was simulated to explain the planner's decision.

Initially, the Simulated World is created from the World DB and then the status of the world is updated as entities are being simulated. The simulator uses the TA to update the Simulated World. The outcome of the simulation is then fed into the COA Tree evaluator.

5. The **COA Evaluator** evaluates the simulation of different plans produced by the Simulator by using measures such as success/failure, number of casualties and equipment loss. The evaluation is done by rules. A point to note however is that the data on which the evaluation is performed is produced through simulations and not rules. Using the evaluation, a plan is chosen based on a combination of user's criteria such as degree of success, minimal loss, randomness. Finally, the Execution Monitor is called with the chosen plan.

6. The **Execution Monitor**

   The Execution Monitor is the main driver of the Planning Behavior module only.

   (a) Issue the set of chosen subtasks in the plan to each units in an Oporder format.

   (b) Execute its own subtask if any.

   (c) If any Sitrep is received,

      i. Call the Sitrep/Oporder Analyzer.

      ii. If the decision returned calls for

         - immediate action, it is handled by the REACTIVE behavior module which accesses the mini Expert System to react accordingly. In doing so, the REACTIVE module also takes into account any Engagement Criteria given in the Oporder.
         - replanning, the SA is called to start a planning process with the newly updated World DB.
         - giving up planning at the current level, the CE sends Sitrep to its higher unit reporting of its current status and waits for further orders.

3.2.4 Expert System

The mini Expert System module contains rules to aid the planning process in making decisions such as choosing routes, choosing best COA tree, performing analysis of situations, Oporders and Sitreps.

4 Future work

Our future work is best divided into two parts: near term and long term. For the near term (next 2-3 months), we are in the process of taking the detailed planner design and implementing it within the IST CGF testbed which is SIMNET-compliant and operates using the IBM PC architecture. The approach will be deemed effective if the trainee learns appropriate maneuvers against the plan-driven CGF adversary. Also, we are simplifying the plan generation in that the enemy's actions are not being fully simulated. The planner architecture, however, allows for concurrent enemy simulation while CGF courses of action are being simulated. In the long term, we plan on adding more capability to the planner once the skeleton architecture has been implemented. We want to run experimental designs on the planner to carefully measure the effectiveness of training. If possible, we would like to measure this quantity against other existing planners. Another key long-term feature of a future planner is machine learning. That is, the planner should be able to observe example engagements to learn rules that can be used to prune the COA data structure. Also, once multiple simulations have been run, the planner should be able to "abstract out" rules which capture the simulation information in a more compact form.

5 Conclusions

We have shown how we are able to perform AI planning with fewer rules by using simulation to project and evaluate potential plans. Simulation allows the planner to project a broader class of results in a uniform manner. In the mission planning aspect, simulation-based planning is useful because it is easily scalable, extendible and explainable. As discussed in section 2, explanation can be done by simply playing back the simulations that were performed internally during the evaluation phase and seeing what led the planner to choose a particular plan. Finally, simulation-based planning makes designing of multi-agent adversarial planning systems more feasible and simplifies the reasoning procedures by allowing many details to be left out and be considered later during the evaluation phase. A drawback of simulation-based planning is that it can be computationally intensive if plans were to be simulated at considerable level of detail. If machine power permits, we can use simulation for all phases of planning; however when we have a limited computing resource, the best practical alternative is to combine rule-based with simulation-based elements as we have done here, or to perform simulation at different levels of abstraction [12].

6 Acknowledgements

This research has been supported by a sub-contract, under the direction of Mikel Petty of IST, to the University of Florida for mission planning. We would like to acknowledge conversations with Clark Karr and Doug Reece of IST. They really helped us to understand many of the issues connected with military planning. Also, we would like to thank Dan Mullally for serving as the resident expert on military maneuvers and terrain reasoning.

References

[1] W. Braudaway. A Blackboard Approach to Computer Generated Forces. In *Proceedings of the Third Conference on Computer Generated Forces and Behavioral Representation*, pages 11–20, Orlando, FL., 1993.

[2] M. Czigler and S. Downes-Martin. Fast Futures Contingency Simulation: A "What If" Tool for Exploring Alternative Plans. In *Proceedings of the 1994 SCS Simulation MultiConference*, San Diego, CA, 1994.

[3] T.L. Dean and K. Kanazawa. Persistence and probablistic inference. Technical Report CS-87-23, Brown University Department of Computer Science, 1987.

[4] T.L. Dean and M.P. Wellman. *Planning and Control.* Morgan Kaufmann, 1991.

[5] R.E. Fikes, P.E. Hart, and N.J. Nilsson. Learning and Executing Generalized Robot Plans. *Aritificial Intelligence*, 3, 1972.

[6] R.E. Fikes, P.E. Hart, and N.J. Nilsson. Some New Directions in Robot Problem Solving. In *Machine Intelligence 7.* Edinburgh University Press, 1972.

[7] P. A. Fishwick. An Integrated Approach to System Modelling using a Synthesis of Artificial Intelligence, Software Engineering and Simulation Methodologies. *ACM Transactions on Modeling and Computer Simulation*, 2(4):307 – 330, 1992.

[8] P.A. Fishwick. *Simulation Model Design and Execution: Building Digital Worlds.* Prentice-Hall, Inc, 1994.

[9] K. Hammond. Cased-based planning. In *Perspectives in Aritificial Intelligence*, volume 1. Academic Press, 1989.

[10] K. Kanazawa and T. Dean. A Model for Projection and Action. In *Proceedings of IJCAI-89*, pages 985–999, Detroit, MI, 1989.

[11] C.R. Karr, R.W. Franceschini, K.R.S. Perumalla, and M.D. Petty. Integrating Aggregate and Vehicle Level Simulations. In *Proceedings of the Third Conference on Computer Generated Forces and Behavioral Representation*, pages 231–239, Orlando, FL., 1993.

[12] J.J. Lee, W.D. Norris, and P.A. Fishwick. An object-oriented mutlimodeling design for integrating simulation and planning tasks. *Journal of Systems Engineering*, 3:220–235, 1993.

[13] E.D. Sacerdoti. *A Structure for Plans and Behaviour*. Elsevier-North Holland, 1977.

[14] M. Schoppers. Universal Plans for Reactive Robots in Unpredictable Domains. In *Int. Joint Conference on Artificial Intelligence*, 1987.

[15] D.E. Wilkins. Domain Independent Planning: Representation and Plan Generation. *Aritificial Intelligence*, 22:269–301, 1984.

7 Biographies

Jin Joo Lee received a B.S. degree in Computer Science from Ewha University, Korea in 1988 and a M.S. degree in Computer Science from Brown University in 1991. After receiving the M.S. degree, she was a research engineer at Human Computers Inc., Korea until 1992. She is a currently a PhD student at the Computer and Information Sciences department at University of Florida. Her research interests are in AI planning, simulation and control.

Paul A. Fishwick is an associate professor in the Department of Computer and Information Sciences at the University of Florida. He received the BS in Mathematics from the Pennsylvania State University, MS in Applied Science from the College of William and Mary, and PhD 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 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.