SF 298 MASTER COPY KEEP THIS COPY FOR REPRODUCTION PURPOSES

REPORT DOCUMENTATION PAGE

Form Approved
OMB NO. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	14 August 1996	Final 1 July 1995 - 30 June 1996

4. TITLE AND SUBTITLE

Combat Simulation Trajectory Management

5. FUNDING NUMBERS

DAAH04-95-1-0350

6. AUTHOR(S)

John B. Gilmer Jr.

7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(ES)

Wilkes University
P.O. Box 111
Wilkes-Barre, PA 18766

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

U.S. Army Research Office
P.O. Box 12211
Research Triangle Park, NC 27709-2211

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

ARO 34433.3-MA

11. SUPPLEMENTARY NOTES

The views, opinions and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy or decision, unless so designated by other documentation.

12a. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

19960909 155

13. ABSTRACT *(Maximum 200 words)*

This project tested the feasibility of explicitly tracking multiple outcomes for random events in combat simulations. A simple force on force combat simulation resembling Eagle was built that creates new states and trajectories when selected random events occur, in order to maximize coverage of the simulation outcome space. This multitrajectory simulation concept was implemented in C++ using a class structure that hides the complexity of spawning new trajectories, so that the functional code programmer's task is nearly the same as for stochastic simulation. The challenge is that a random choice method called once must return multiple times, once for each trajectory, if an event is resolved in a multitrajectory fashion. The prototype simulation was tested for 4 unit and 40 unit scenarios, with a variety of management strategies selected to manage (and ration) trajectory proliferation. The results showed superior information gained for a given commitment of computational resources, compared to stochastic simulation. There remain some software issues in the use of local variables. The project was unable to address scaling issues to larger sized scenarios, which will require more sophisticated management heuristics.

14. SUBJECT TERMS

Simulation, stochastic, random events, simulation trajectory, trajectory management, combat simulation.

15. NUMBER IF PAGES

29

16. PRICE CODE

17. SECURITY CLASSIFICATION OR REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

NSN 7540-01-280-5500

Enclosure 1

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

REPORT DOCUMENTATION PAGE (SF298)
(Continuation Sheet)

LIST OF MANUSCRIPTS

John B. Gilmer Jr. and Frederick J. Sullivan, "Combat Simulation Trajectory Management", Proceedings of the Military, Government and Aerospace Simulation Conference, 1996, The Society for Computer Simulation, April 1996, pp 236-241. This paper reported early results.

Frederick J. Sullivan and John B. Gilmer Jr., "Managing Multiple Trajectory Parallel Simulation", Proceedings of the 1996 High Performance Computing Conference, The Society for Computer Simulation, April 1996, pp 320-323. This paper reported early results.

John B. Gilmer Jr., Frederick J. Sullivan, and Sadeq Al-Hassan, "Testing of Multitrajectory Techniques for Military Simulation", was presented in Working Group 33, Modeling, Simulation, and Wargaming, of the 64th Military Operations Research Society Symposium, June 1996, and submitted DTIC for publication.

Frederick J. Sullivan, John B. Gilmer Jr., and Sadeq Al-Hassan, "Software Issues in Multitrajectory Simulation" was presented to Working Group 31, Computing Advances in Military Operations Research, of the 64th MORS Symposium, June 1996, and submitted to DTIC for publication.

SCIENTIFIC PERSONNEL

John B. Gilmer, Jr., Principal Investigator

Frederick J. Sullivan, Investigator

Sadeq Al-Hassan, Graduate Research Assistant. He is expected to earn an MSEE in January, 1997, with a thesis based on work done on this project.

Report of INVENTIONS: none

SCIENTIFIC PROGRESS AND ACCOMPLISHMENTS:

Demonstrated that the multitrajectory simulation technique can be practically implemented. Also, showed that this technique adds value for force on force combat simulation. For a given commitment of computational resources, the multitrajectory technique is superior to stochastic simulation in giving the analyst an understanding the outcome space.

TECHNOLOGY TRANSFER:

In addition to presentation of papers at conferences, gave briefings to the Head of the U.S. Army Concepts Analysis Agency, Deputy Undersecretary of the Army for Operations Research, and to two private companies, BDM International and Pioneer Decision Technologies. The Final Report briefing delivered is an appendix to this report.

COMBAT SIMULATION TRAJECTORY MANAGEMENT

FINAL REPORT

JOHN B. GILMER JR.

14 AUGUST, 1995

U.S. ARMY RESEARCH OFFICE

GRANT DAAH04-95-0350

WILKES UNIVERSITY

TABLE OF CONTENTS

LIST OF APPENDIXES, ILLUSTRATIONS, AND TABLES

STATEMENT OF THE PROBLEM

The management of random outcomes in stochastic simulation, coupled with the option to track multiple outcomes of stochastic events, was proposed as a way of obtaining more information about the outcome space of a combat simulation than could be achieved by random stochatic choices as is normally done. This project was intended as an exploration of this concept to see whether the technique could be practically implemented, and whether it would achieve worthwhile benefits.

The problems included both implementation issues and usefulness issues. Creating additional trajectories for a simulation at random events involves potentially multiple returns from the random choice mechanism for each call. A class and object structure was sought to hide the multitrajectory implementation behind a facade that would allow programming as if the simulation were conventionally stochastic. This proved a more difficult challenge than expected.

The second issue was whether the management techniques used to prevent unbounded combinatoric explosion in the face of many events would still deliver results that were more useful, given the computational resources used, than would stochastic simulation. This issue was explored in the context of a force of force simulation, with Eagle (a simulation used for analysis at the U.S. Army Concepts Analysis Agency) being the exemplar to provide a context.

These issues were pursued by building a simple prototype simulation "eaglet" in C++ that resembled Eagle in certain essential features, implementing the class structures and features to support multitrajectory simulation, and exercising the simulation for a small scenario about the scale of the smaller Eagle scenarios.

SUMMARY OF THE MOST IMPORTANT RESULTS

We found that it is indeed possible to encapsulate the multitrajectory features in a class structure in C++ that allows the programmer to develop functional model code as if it were stochastic. However, there were problems in the use of local variables when the state context changes on the second return from a "chooser" (that resolves the random event) in a multitrajectory event. The most troubling aspect became known as the "this" problem. In C++ the local variable "this" points to the current object. Upon return from a multitrajectory event, "this" incorrectly points to the current object, but in the wrong trajectory. This cannot be easily fixed because in C++ "this" is not writable. A discipline was developed that allows these problems (including the "this" problem) to be avoided. A cleaner interface would be desirable, but could not be developed within the scope of this project.

The multitrajectory technique, with a variety of choice management options, was explored in the context of a very simple four unit scenario and a larger scenario of about 40 units. The smaller scenario allowed exhaustive comparison of all possible outcomes, which was particularly useful for verification. It also gave encouraging results for cases where resources are available to cover a large proportion of the outcome space, showing that the multitrajectory approach gives more information about the outcome space than larger amounts of computation applied to randomly determined trajectories.

For the mid-sized scenario, results were less conclusive, because the scenario could never be made to operate correctly on the larger target machines. Consequently tests were limited to 800 states, which was a small proportion of the outcome space. Nevertheless, a

combination of multitrajectory and stochastic event resolution strategy gave better results than a purely stochastic simulation with the same number of trajectories. These tests pointed out the need for more research in the trajectory management heuristics, and the issues of scaling in general.

A more detailed report of results can be found in Appendix A.

LIST OF PUBLICATIONS AND TECHNICAL REPORTS

PUBLISHED OR PRESENTED REPORTS

John B. Gilmer Jr. and Frederick J. Sullivan, "Combat Simulation Trajectory Management", Proceedings of the Military, Government and Aerospace Simulation Conference, 1996, The Society for Computer Simulation, April 1996, pp 236-241. This paper reported early results.

Frederick J. Sullivan and John B. Gilmer Jr., "Managing Multiple Trajectory Parallel Simulation", Proceedings of the 1996 High Performance Computing Conference, The Society for Computer Simulation, April 1996, pp 320-323. This paper reported early results.

John B. Gilmer Jr., Frederick J. Sullivan, and Sadeq Al-Hassan, "Testing of Multitrajectory Techniques for Military Simulation", was presented in Working Group 33, Modeling, Simulation, and Wargaming, of the 64th Military Operations Research Society Symposium, June 1996, and submitted DTIC for publication.

Frederick J. Sullivan, John B. Gilmer Jr., and Sadeq Al-Hassan, "Software Issues in Multitrajectory Simulation" was presented to Working Group 31, Computing Advances in Military Operations Research, of the 64th MORS Symposium, June 1996, and submitted to DTIC for publication.

In addition, briefings were given to Mr. Vandiver, Head, U.S. Concepts Analysis Agency, and Mr. Hollis, Deputy Undersecretary of the Army for Operations Research. An annotated copy of the latter briefing is included in Appendix A.

UNPUBLISHED REPORTS, AVAILABLE ON REQUEST

John B. Gilmer Jr., "Combat Simulation Trajectory Management, Preliminary Simulation Specifications", August 14, 1995.

John B. Gilmer Jr., "Combat Simulation Trajectory Management Development Issues", August 26, 1995.

John B. Gilmer Jr., "Metrics for Eaglet State Management", Sept 18, 1995.

John B. Gilmer Jr., "Managing this Forking Program", Sept 19, 1995.

John B. Gilmer Jr., "Eaglet Operations and Rules Summary", Nov 10, 1995.

John B. Gilmer Jr., "Sampling for Eaglet Attrition", Nov 17, 1995.

John B. Gilmer Jr., "Outcome Space Characterization", Jan 22, 1996.

John B. Gilmer Jr., "Outcome Space Characterization and Analytic Assessment Techniques", February 12, 1996.

Note: These unpublished reports were primarily used as design documentation and as a vehicle for communicating features needed for the simulation within the project team.

LIST OF ALL PARTICIPATING SCIENTIFIC PERSONNEL

John B. Gilmer, Jr., Principal Investigator

Frederick J. Sullivan, Investigator

Sadeq Al-Hassan, Graduate Research Assistant. He is expected to earn an MSEE in January, 1997, with a thesis based on work done on this project.

REPORT OF INVENTIONS none

BIBLIOGRAPHY

LTC Robert S. Alexander, Briefing on Eagle received on 19 July, 1995. The information received is documented in a memo by J.B.Gilmer of July 21, 1995. This briefing was the basis of the functional design for the prototype simulation "eaglet" used in this project.

"Eagle Combat Model", TRADOC Analysis Command -- Operations Analysis Center, Ft. Leavenworth, Kansas. This is a copy of a presentation on Eagle describing its architecture and functionality.

John B. Gilmer Jr., "The Effects of Decisionmaking Quality and Timeliness on the Response Surface of a Simple Combat Simulation", Paper presented at the 63rd MORS Symposium, June 1995. This paper describes prior research performed on combat simulation nonmonotonicity, with a focus on C2 effects.

John B. Gilmer Jr. and James A. Adams, "Managing Uncertainty Explicitly in Simulation", Paper presented at the 62nd MORS Symposium. The concept described in this paper was the genesis of this project. This paper was included in the project proposal.

Bodhisattwa Mukherjee, Greg Eisenhauer and Kaushik Ghosh, "A machine Independent Interface for Lightweight Threads", Operating Systems Review of the ACM Special Interest Group in Operating Systems, pp 33-47, Jan 1994. This paper discusses Cthreads, which was one option considered for implementing the continuations of events.

William Klinger and Jonathan Rees, "Revised^4 Report on the Algorithmic Language Scheme", describes continuations in the Lisp dialect Scheme, which was closer in function to the capability needed in this project.

APPENDIX A Annotated Final Report Briefing

Combat Simulation Trajectory Management

Final Report

25 July 1996

John B. Gilmer Jr.
Frederick J. Sullivan

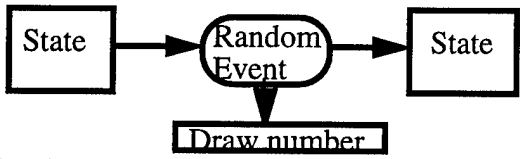
Department of Electrical and Computer Engineering
Department of Mathematics and Computer Science

Wilkes University
Wilkes-Barre, PA 18766
(717) 831-4885

This report was delivered to the Deputy Undersecretary of the Army for Operations Research, the Hon. Walt Hollis. A somewhat more detailed version was delivered to Mr. Vindiver, Head, U.S. Army Concepts Analysis Agency on 16 July.

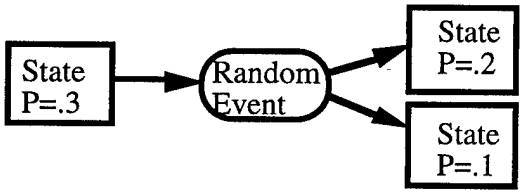
The Multitrajectory Concept

Conventional Simulation:



Each replication gives only one outcome, randomly determined

Multi-Trajectory Simulation:



Each replication gives numerous outcomes, characterized by their probabilities

The principle behind this project is the generation of multiple trajectories from random events. The word "event" is used for the point in a simulation at which a random choice may be made, resulting in possibly different outcomes. This is not identical to the meaning in the sense of "event sequenced" simulation. If the probabilities of the various outcomes of an event are known, one can keep track of the probabilities of the various trajectories.

Motivation for Multitrajectory Simulation

Simulation needs to **represent** the real world,
but need not **operate like** the real world.

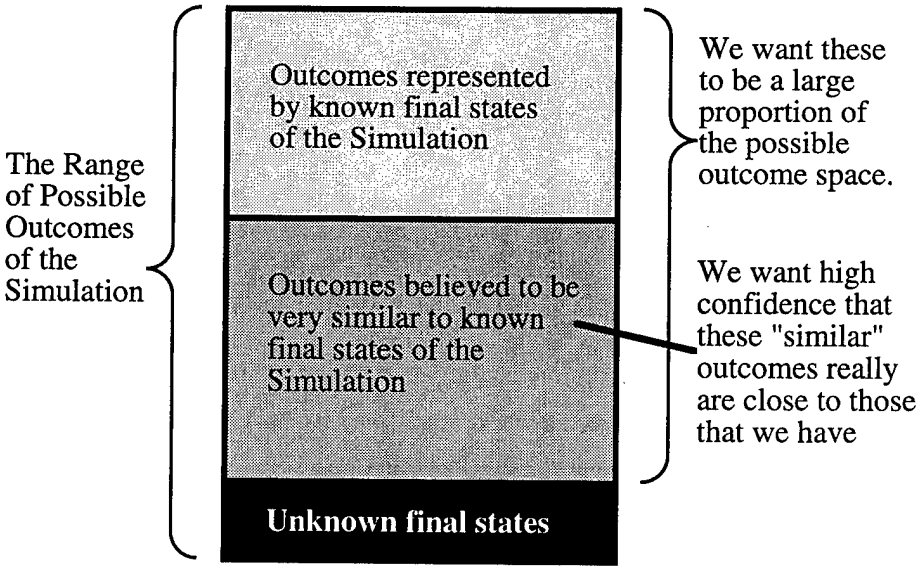
Real world: Only one trajectory

For the Analyst: We would like to understand
all of the possible trajectories
and their probabilities

Motivation: Treat simulation as a tool to address the analyst's problem rather than one that operates like the real world.

We have also been concerned with the effects of "chaos" in simulation response, and would like a tool that would be better able to generate response surfaces that cover a multitude of parameters and events more efficiently.

What we hope an Analyst will see at the end of a simulation "run"



The outcome space can be thought of as the full range of possible outcomes given randomness in a defined set of events. For those events having randomness that most affects the outcome, we would like to have good coverage in our set of final states. Where we do not have the actual final states, we would at least like to know that in most cases those states we did not retain were similar to those we did follow. There will usually also be those that were not followed, perhaps because they were each so improbable, though numerous, that the resources to follow them did not seem worthwhile.

Background: Origins of this Project

1. Determinism vs Stochastic nature of Combat
   (Discussions during design earlier combat models, early 80's)

2. Analysis of "Great Battles" wargame combat system-
   Design and implementation of exhaustive analyis software

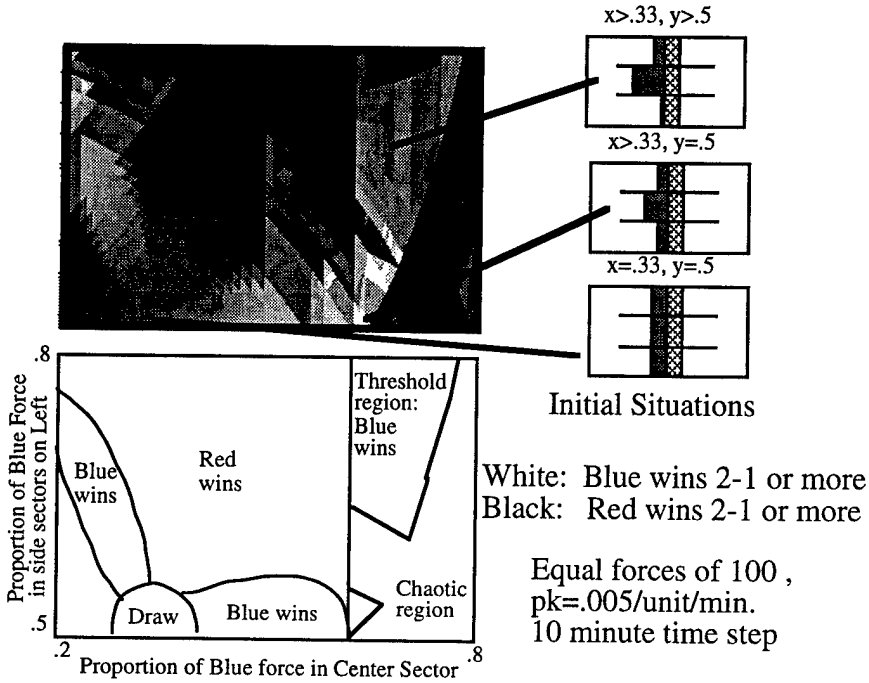
3. Concern for analyst's perspective tracing back to Simtech 97

4. "Chaos" as way of understanding sensitivity of combat models-
   Design and use of "CombatChaos" model to demonstrate this

Conclusion: Knowing that combat models are chaotic does not solve problems . What is needed is a way to manage their operation to produce useful results in the presence of chaotic behavior. Motivation for this effort.

A number of projects and reports have identified chaos or nonmonotonicity in combat simulations as a possible pitfall for analysis. Unfortunately, knowing there is possible chaotic behavior is not the same as knowing what to do something about it.

Chaos Observed in 3 sector Lanchester Model



This is an example of such a study, that revealed nonmonotonic responses of a simple simulation as initial Blue dispositions vary. The Blue disposition parameters are only two of a host of parameters and event outcomes that can be thought of as random. Parametric studies such as this are inadequate to address all of these possible sources of variation.

Combat Simulation Trajectory Management

Funding: From CAA via ARO, $47,615
Duration: 1 year (July 95 to June 96)

Staffing: Principal investigator (J. Gilmer) 1/4 time
Investigator (F. Sullivan) 1/12 time
Graduate Research Assistant 1/2 time

Objective: Examine experimentally the concept of Multitrajectory Simulation and determine its practicality and benefit for large scale combat simulation

We hope to show that by explicitly controlling the treatment of probabilistic events, we will convey more information to the analyst at lower cost than with multiple stochastic replications. Furthermore, we expect to show that this is practical in terms of software development.

Combat Simulation Trajectory Management Project Tasks

Task 1.1 Define objectives and characteristics for the surrogate simulation.

Task 1.2 Develop the prototype simulation.

Task 1.3 Perform initial replications to characterize the simulation.

Task 2.1 Assess the simulation's behavior against the prototypes.

Task 2.2 Assess simulation trajectory behaviors under a stochastic distribution of critical event outcomes.

Task 2.3 Implement an Initial trajectory management technique based on similarity and pruning.

Task 2.4 Evaluate the effectiveness of proposed management techniques

As the project developed, and due to the availability of Dr. Frederick J. Sullivan, it was possible to focus more on implementation issues than originally expected. Thus, we were able to develop a much more comprehensive foftware approach than originally anticipated. Unfortunately, the new prototype could not be made to operate on the Silicon Graphics machines of the Wilkes Simulation Lab, so that the larger scale tests could not be carried out.

Issues and Approaches

The basic problem: Exponential explosion
in the number of states as the number
of random events grows large.

Suppose each random event has only two possible outcomes:
After only 10 events, 1024 states have been created.
The average number of states is 205. Both numbers grow rapidly.

Possible ways to control this problem:

1. Limit the number of random events
   Reserve explicit treatment of random trajectories
   to "critical" events that result in divergent trajectories

2. Consolidate identical states, Represent state differences
   These are software techniques for gaining efficiency

3. Consolidate similar states at some cost in Confidence
   Only the most probable, or most representative, states are
   tracked explixcitly.

The Surrogate Simulation: "eaglet"

Intended to resemble Eagle in functional characteristics

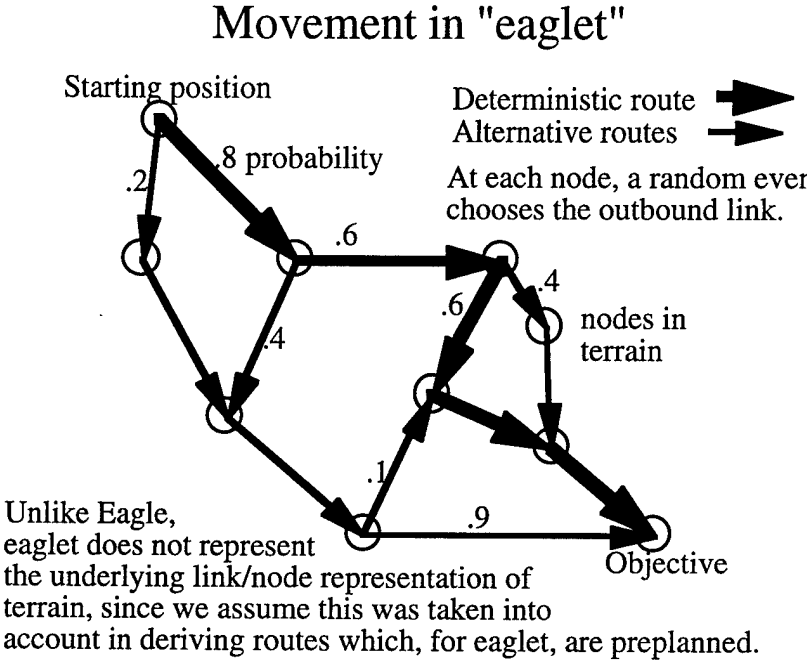
Intended to be <u>much</u> simpler than Eagle

Written in C++ (due to facilities for handling objects, etc.)

Functional Characteristics:
1. Link-Node movement along preplanned routes
2. Units have "strength" rather than list of weapons
3. Attrition does reflect front, flank, rear sector allocations
4. Aquisition loss/gain on enemy units can be stochastic
5. Units each have "Task" with objective, intent, etc.
6. Rule based decisionmaking
7. Time stepped sequencing

Initial version: Multitrajectory resolution for 5 events:
Movement selection, attrition, Acquisition gain/loss,
decisionmaking

The approach in this project was experimental, but the complexity of the simulation software needed to stay bounded for reasons of practicality. The original "Eagle" is written in Lisp, but we wished to stay with C++ because of its object handling and availability. The representation of functional characteristics in "eaglet" was kept similar to those in Eagle, but with great simplification. For example, instead of having a list of items, each unit has only abstract units of "force". Where in Eagle units plan routes, in eaglet all routes were pre-planned. This allowed better control of the project and managable verification.

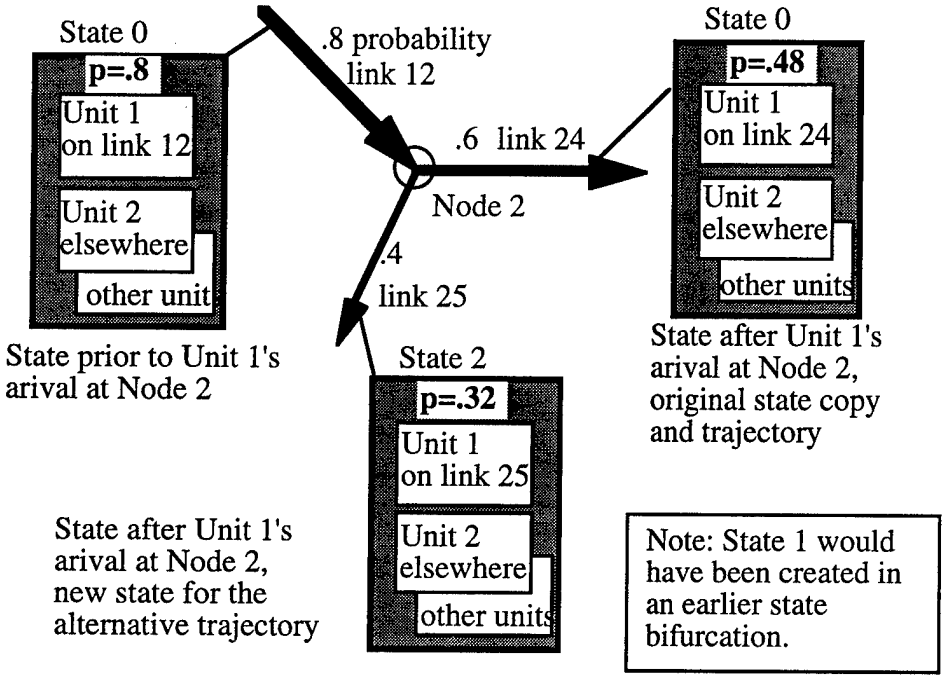


Eagle is deterministic, while we needed events in eaglet to represent stochastic processes. The "movement selection" event is used to illustrate. In Eagle, a unit would plan a route consisting of a number of nodes connected by a single chain of links. A random process (if implemented) for generating the route might choose one of a number of different possible routes. To avoid having to build a planner, routes in eaglet represent the network of possible routes that a stochastic path planner might generate. This structure, as illustrated above, is an input. Because it is an input, for this event we can predetermine the possible outcomes and their probabilities, which is helpful for verification of the multitrajectory event handling software. (This would not be possible for some other events.)

For a unit starting at the point in the upper left corner, there is a .8 chance the unit would follow the heavy arrow, and a .2 chance it would go the other way. At other points there are other possibilities for random choices. This particular route can be traversed 6 different ways. If there were four units following similar routes, and all were resolved with multitrajectory events, there would be 1296 possible trajectories if this one event is always resolved in a multi-trajectory fashion.

However, the event could be always resolved deterministically, by always taking the most probable path (heavy arrows). Or, as with stochastic simulation, random choices could be made. Or, and this is a focus of ongoing research, the way in which such events are resolved is managed, to give good coverage ofg the outcome space, while preserving at least some of the important variability of interest.

Multitrajectory Movement Events

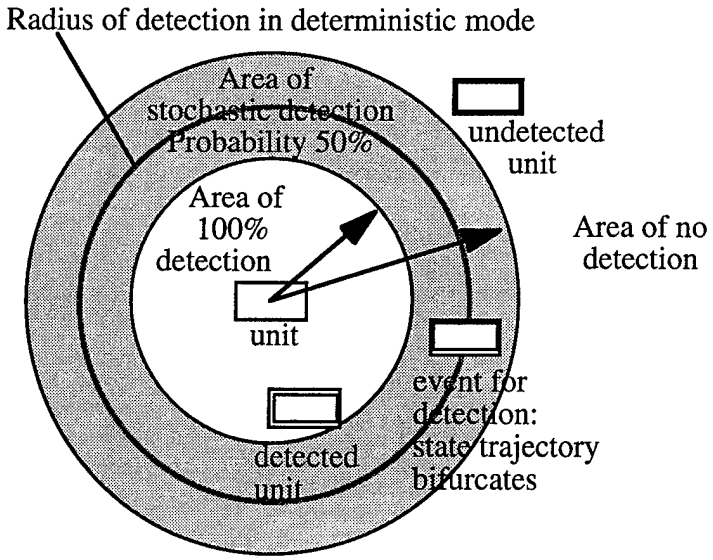


Here we see from a software perspective what happens when a movement selection event is resolved in a multitrajectory fashion. At the time the event is recognized, we are in the context of a particular trajectory, or state, in this case State 0. Presumably State 1 already exists (and possibly others), since State 0 has a probability of .8 rather than 1. In this trajectory, Unit 1 has arrived at a point where a choice must be made between two different paths, following link 24 or link 25. Since we are resolving this event as multitrajectory, both outcomes occur. The original state follows the more probable trajectory (which would have been the deterministic choice). We see, though, that the probability of State 0 is now changed to .48, reflecting the accumulated probability of being in that trajectory given the events so far encountered.

In addition, a new trajectory, and its state, must be created. In this new trajectory, Unit 1 takes the other path instead, Link 25. The sum of the trajectories' probabilities is equal to that of the trajectory coming into the event. The state variables associated with other units in the trajectory would remain unchanged. Note that such a trajectory bifurcation occurs for every event, regardless of which unit encounters the event.

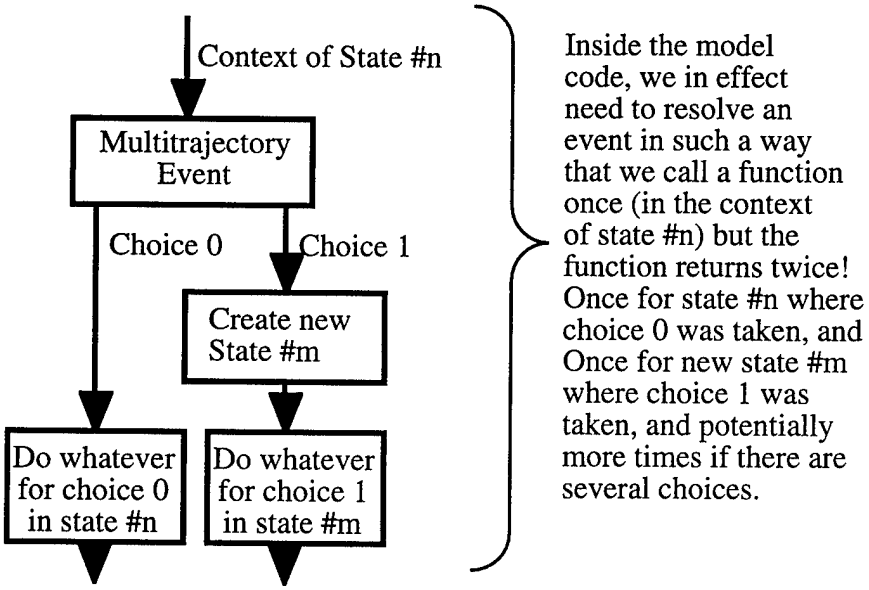
The set of events for which multitrajectory treatment was provided was chosen to cover a wide variety of cases typical of those which one might find in an analytic combat simulation. The movement selection event shown here has varying numbers of outcomes, and the probabilities vary with the particular event. Most other events are simpler.

Acquisition in "eaglet"



Acquisition is a binary choice with a constant parameter of .5, the simplest case. Note that the algorithm is constructed so that the deterministic choice uses the mean of the range limits used for the stochastic case.

What Has to Happen



Inside the model code, we in effect need to resolve an event in such a way that we call a function once (in the context of state #n) but the function returns twice! Once for state #n where choice 0 was taken, and Once for new state #m where choice 1 was taken, and potentially more times if there are several choices.

Even in this simple case, the software problem gets messy. Somewhere in the bowels of the code, a "chooser" routine is called to make a choice. In the multitrajectory case, that single chooser call needs to generate two different returns, one for each choice. This is not conventional software! This violates a rather fundamental way we expect software to behave, and yet this is exactly what we need in order to generate new trajectories cleanly.

Software Issues:
How not to manage state bifurcation

(code buried inside functional module of first prototype)

```
if(lose_acq_evt==0){ /* deterministic */
   Acquisition_list[i]=0;
   N_acquired--;}

else if(lose_acq_evt==1){ /*stochastic */
  rand_num=rand()/32768.0;
  if(rand_num<pct_lose){
    Acquisition_list[i]=0;
    N_acquired--;}}

else if(lose_acq_evt>=2){ /*multiple */
  if(p_state-
>status_event()==LOSE_ACQ_EVT&&
       p_state->status_unit()==Id&&
       p_state->status_itteration()==i){
    Acquisition_list[i]=0;
    N_acquired--;
    p_state->create_status(0,Id,0);}
/*reset status*/
   else{
     p_new_state=new
State(p_state,pct_lose);}}
```

Somehow, the new state has to reenter and get to this point in the code.

The initial prototype embedded code in the functional routines to handle the creation and initialization of new trajectories. It was messy. A lot messier than what you see here, since there needs to be a thread of conditional statements that gets you back to this point where the new trajectory was generated. The code also gets much more messy if you want a variety of management strategies. This was a very early, relatively simple version used in the initial prototype before the new class libraries were developed.

(See figure below) Here we see how a stochastic simulation writer might be inclined to code a random choice, and how it needs to be done to support multitrajectory simulation. The new multitrajectory class structure developed by Fred Sullivan hides the messy details needed in the first prototype. There are differences from how a purely stochastic simulation would do things. The most important difference is that the choice mechanism needs to understand that what is being done is a binary choice, rather than a continuous choice. A random number generator knows less about how its result is used than is necessary to support the multitrajectory mechanism. Our objective was to make it relatively easy to retrofit multitrajectory techniques into existing simulations.

Differences in How to Write Code for Random Events

```
r = random();
if(r < .6){
    do_it(this);}
```

Note that the called random number function does not know that the supplied number will simply be used to make a binary choice.

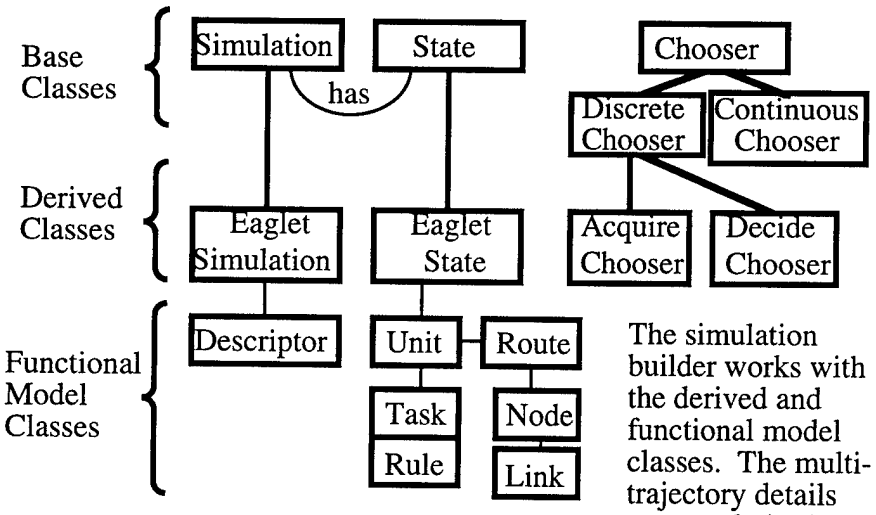
```
c = binary_choice(.6)
if(c==0){
    do_it(this);}
```

(This is the goal. In fact, it is not quite this simple.)

Here we express to the called function that we are just interested in two possible outcomes. This is what the functional programmer needs to do to best support multitrajectory simulation.

The example above is the simplest type of multitrajectory event, a fixed probability binary choice, with a probability of doing something (by a call to "do_it") of 60%. The "this" variable is the C++ local variable containing the pointer to the current object, typically a military unit entity. One of the problems with C++ (for our purposes) is that this "this" variable cannot be modified in the code. On the first return from binary_choice, c will be zero and "this" will be unmodified: it still points to the same instance of the object as when binary_choice was called, since the first return is in the context of the same trajectory, and state. However, on the second return from binary_choice, the "this" variable will still point to the instance of the object which is in the original trajectory, not the current (c=1) trajectory. The preferred solution is to overwrite "this" on the way back from random_choice, but it is not possible to do so in C++. An alternative fix is to not use

Objects



The simulation builder works with the derived and functional model classes. The multi-trajectory details are mostly in the base classes.

Note: Avoid pointers as state variables, use ID's instead.

"this", and substitute another local variable "self" instead where "this" would normally be used. In this particular case, there really is no problem, since "this" is not used when c=1.

The object structure puts the actual handlers of events, including the adjustment of probability, cloning of states, and such into the base classes that provide the essentials of multitrajectory simulation. The functional code writer, who is actually programming the representation of movement, attrition, and other simulation model functions, would interact with choosers and would have to know very little about the base classes. The simulation and its states would also be objects, derived from the base classes. There would have to be, for the derived state class, a method that would create the clone of a state. If the functional coder needs choosers of a type not provided in the base classes, it could be derived, as was done for the "decide" chooser that selects whether a C2 rule fires or not.

In addition, choice policies for each event type would be controlled by methods that use a standard list of possible policies (e.g. always determiunistic, always stochastic, or multitrajectory until n states, then deterministic). The choice policy methods could be overridden by custom designed policies, for example ones sensitive to Measures of Effectiveness of concern in the particular model.

One restriction is made on the state variables in the state: pointers should not be used. Instead of a pointer, use an identifier, e.g. Unit #43 rather than the unit located in memory at address 0xffa054e0. This is good practice for other reasons of memory efficiency, debugging ease, and a necessity if distributed computing is to be possible.

Other Multitrajectory Events:

Acquisition Loss: Never within range of possible detection
Possible to some outer radius
Always outside the outer radius

Attrition: High, Median, and Low values for losses to a unit during a given time step
OR: High and Low losses (2 vice 3 outcomes)

Decisionmaking: Rules are evaluated with low, median, high criteria for: Effectiveness, Being at/ closing/ approaching objective.
Probability of rule firing depends on how many of these criteria are met:
All: 100% two: 80% one: 40% none: 0%

As mentioned earlier, a variety of event types were selected for multitrajectory treatment, in order to ensure the technique would be able to cover a range of different kinds of random results. Perhaps the most important is acquisition gain or loss. This is the event most easily modeled as stochastic.

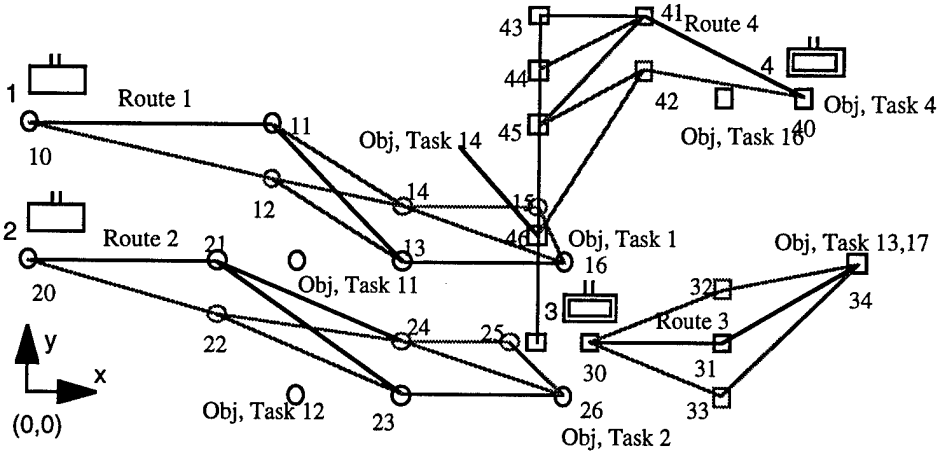
Attrition is unique among those events treated in that a stochastic resolution is inherently more faithful to reality than multitrajectory can be, as it is a case of a continuous chooser. The stochastic choice could be any value within some continuous range. It is not possible to represent all of the possible outcomes, so some representative outcomes (a few samples)

must suffice, for example the mean and plus or minus one standard deviation. As it turned out, attrition events were not nearly as important in generating diverging trajectories as the others examined, so this event was not examined in detail in our later testing.

The decisionmaking event concerns whether a rule fires or not. A rationale for assigning probabhilities for various rules (perhaps depending on circumstances) was beyond the scope of this project.

A Simple Test Scenario

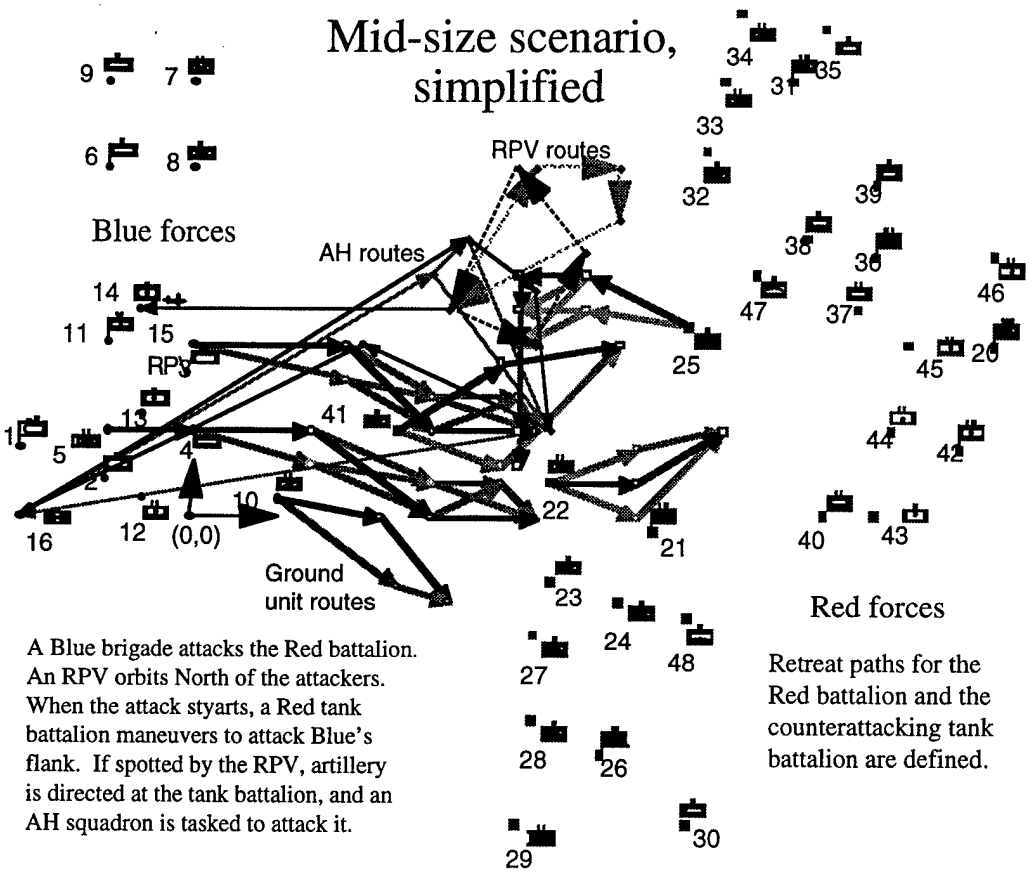
Two Blue units attack one Red unit, Another Red unit counterattacks



Units 1 and 2 attack abreast toward defensive position held by Unit 3. Tasks 11 and 12 are withdraw operations to be executed if either looses effectiveness.
Unit 3 is to hold its initial defensive position. If it loses effectiveness, it is to delay (Task 13) or withdraw toward a point to its rear.
Unit 4 is to wait until Unit 3 is in combat. It is then to maneuver onto the flank of the Blue units and to attack southward. If it loses effectiveness, it is to withdraw toward the NE. If, at its initial objective, it finds no targets, it is to continue attacking South.

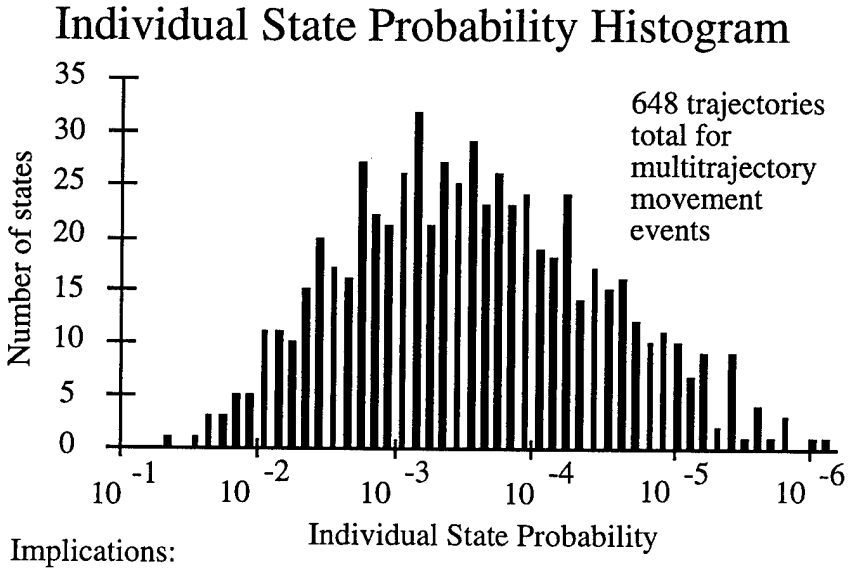
The scenario used for initial testing included only four units. Yet it is representative, to the simplest extent possible, of the more interesting Eagle scenario on which our scenarios were based. The two Blue units attack, and one Red unit attacks on their flank, resulting in some interesting maneuver. Resolved deterministically, the battle results in Blue being thrown back by the counterattack, with heavy losses on both sides. With multitrajectory movement (only), this senario generates 648 trajectories, which proved to be a very managable number for debugging, verification, and initial analyses.

The mid-sized scenario (see following figure) is similar to an Eagle scenario used in a CAA study of non-lethal weapons reported by LTC Maymont at the 63rd MORS Symposium. In it Blue has a Remote Piloted Vehicle (RPV) which flies around looking for potential problems on Blue's flank. If it sees the Red counterattack force, an attack helo squadron is called in. The figure does not show the subsequent attack by the other Blue brigade and other reactions by Red units in the interest of clarity. Most of the testing did not include those subsequent operations. This scenario was never able to run successfully on the Silicon Graphics computers of the Wilkes Simulation Lab, limiting the scope of testing that could be done within project time and budget constraints.



A Blue brigade attacks the Red battalion. An RPV orbits North of the attackers. When the attack styarts, a Red tank battalion maneuvers to attack Blue's flank. If spotted by the RPV, artillery is directed at the tank battalion, and an AH squadron is tasked to attack it.

Retreat paths for the Red battalion and the counterattacking tank battalion are defined.



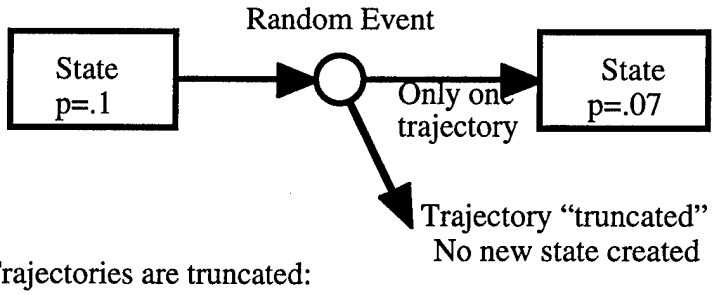
Implications:

10% of the states account for 63% of the outcome space
29% of the states account for 86% of the outcome space
40% of the states account for 96% of the outcome space

When the small (4 unit) scenario is executed with multitrajectory movement events, we get a histogram of state probabilities as shown above. This suggests we can get very good outcome space coverage with a small proportion of the states by "truncating" the less

probable states. This is done by never creating those trajectories. Events for trajectories that are low in probability are "truncated" by resolving them in deterministic (or stochastic) fashion rather than by the multitrajectory technique. This results, of course, in incomplete coverage of the outcome space; the truncated states are "unknown".

Trajectory Truncation:



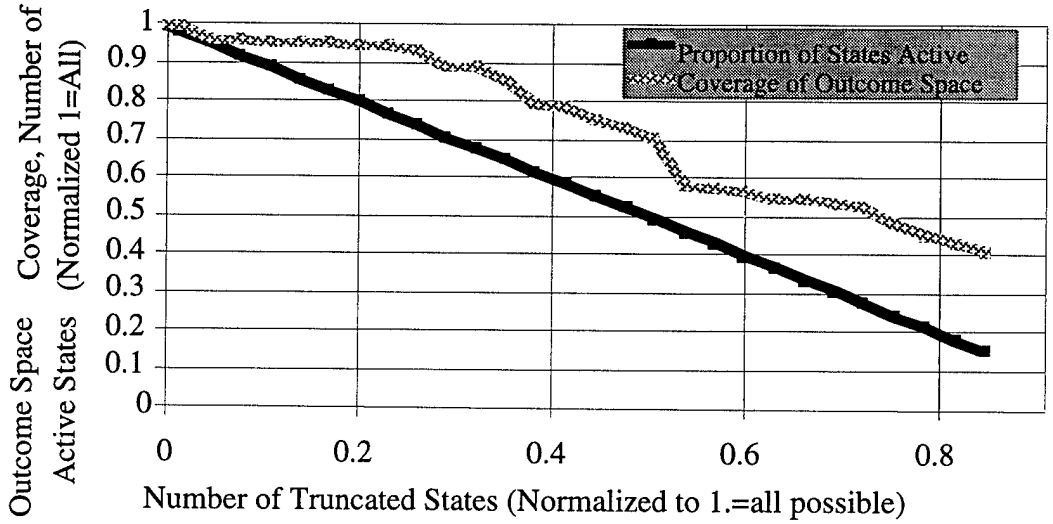
Trajectories are truncated:

When the probability of the truncated trajectory is small enough, and perhaps its metrics unimportant enough, to be worth expending resources for computing it.

The outcome of the event is deterministic or random

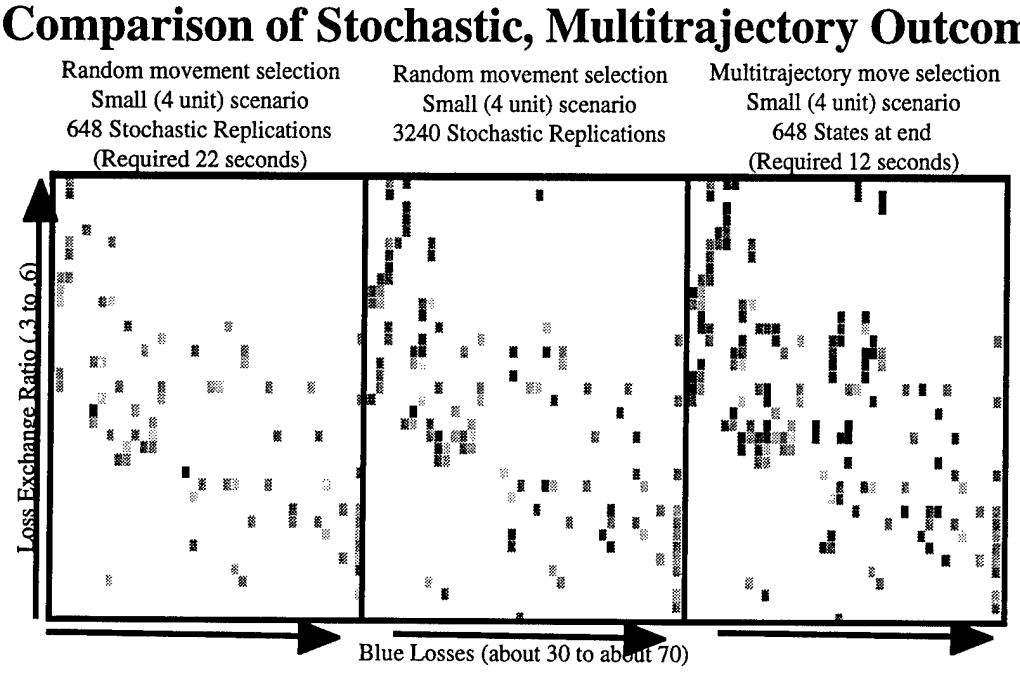
Resources are saved, but some proportion of the outcome space (which the truncated state would reach) remains unknown.

Outcome Space Coverage vs Proportion of Truncated States



Here we see the results of truncating various numbers of states. On the left, we keep all 648. To the right, the proportion of that number decreases. The impact on the coverage is very small at first, then slopes downward significantly. The approach taken to generate

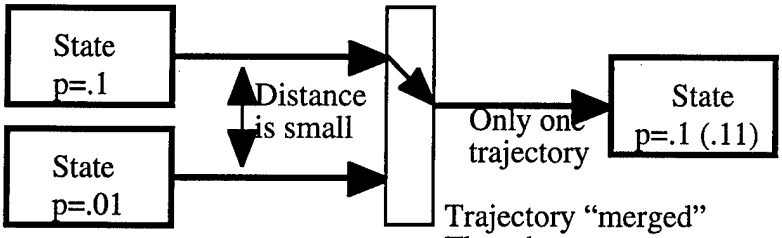
this graph was a very simple choice policy: The choice policy #3 used to generate this data is to use multitrajectory techniques until a defined maximum number of states is reached, then resolve all events deterministically. A policy that was sensitive to trajectory probability might have given a smaller slope to the coverage statistic. Time did not permit a closer examination. The overall effect would be to chop off the lower part of the curve shown in the probability histogram. However, this choice policy does not do that, ti simply stops generating new trajectories at some point, with the consequence of narrowing the distribution and shifting it left a bit. There are limits to savings by truncating states, as the resulting probability distribution has less and less "tail" of low probability states. Taken to its extreme, all states would have about the same probability.



Here we see the impact of random versus multitrajectory choices for the movement event, as portrayed as a distribution over an outcome space. The outcome space is plotted in terms of Blue loss and Loss Exchange Ratio, with the darkness of the squares being the relative probability of states falling in that box. The most probable box in each plot is blac, and less probable ones are lighter colors. (Later in the project, and absolute scale was adopted, which proved more useful.) For this case, the multitrajectory plot is exhaustive; it includes all possible outcomes. Note that it is both faster, and gives a more comprehensive set of outcomes, than a similar number of random trajectories. It even does better than five times the number of random trajectories. The reason a multitrajectory run for a given number of states is quicker than stochastic simulation is the fact that the trajectories are not generated until well into the run; initially there is only one trajectory. The choice policy influences the growth rate of the number of states, and hence the relative speeds given a particular number of final states.

(See next figure) Another way of reducing the number of trajectories is to "merge" those which are similar. This gives some assurance that a "merged" trajectory was similar to one that was kept, so this technique is preferred (though quite a bit more expensive) than truncation. In our testing, checks to determine state mergers were performed every 25 minutes of simulated time rather than at 5 minute stime steps used for model functions.

Trajectory Merging:



Trajectories are merged:

Trajectory "merged" The other represents both.

When the probability of a trajectory is small enough, and its metrics close enough to some other state, the state is purged and the neighbor continues and represents both.

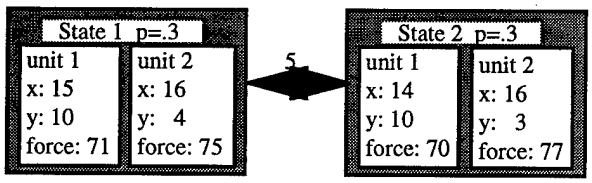
Resources are saved, but some proportion of the outcome space (which the truncated state would reach) remains only "expected" to be similar to the known outcome. Merging is expensive in the cost of comparing states.

Characterizing the Outcome Set

1. Metric for "who wins" etc.: the typical kinds of MOE used for simulations, e.g. loss rate, movement

2. Metrics characterizing the nature of the outcome set:

   a. Distance: the (weighted total) distance between states



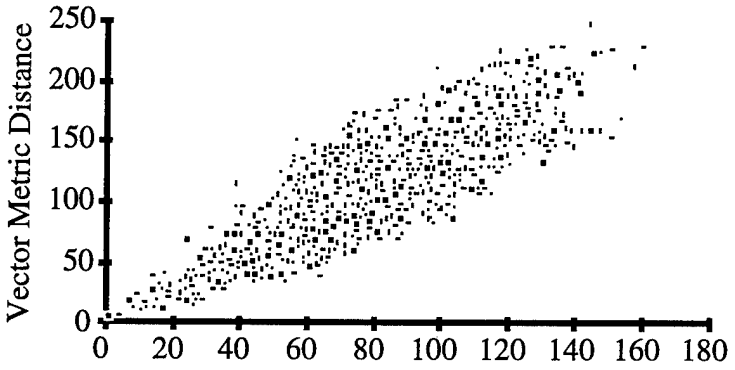
   Distance = $\Sigma |x_i - x_j| + \Sigma |y_i - y_j| + \Sigma |force_i - force_j|$ .....
   with summations over all units, for states i and j

   b. Metrics that are computationally cheaper as a surrogate for the actual distance. (This still needs work)

If merging is to be done, we need to be able to judge when to states are "close" to each other. This requires some sort of metric, since an exhaustive comparison of the actual distance, a sum of the differences in all the state variables between two states, would be prohibitively expensive. It took a vector of indicators to give a metric that would adequately characterize a state, for the purpose of deciding when two states could be merged. No doubt more research could result in finding a better, less expensive metric.

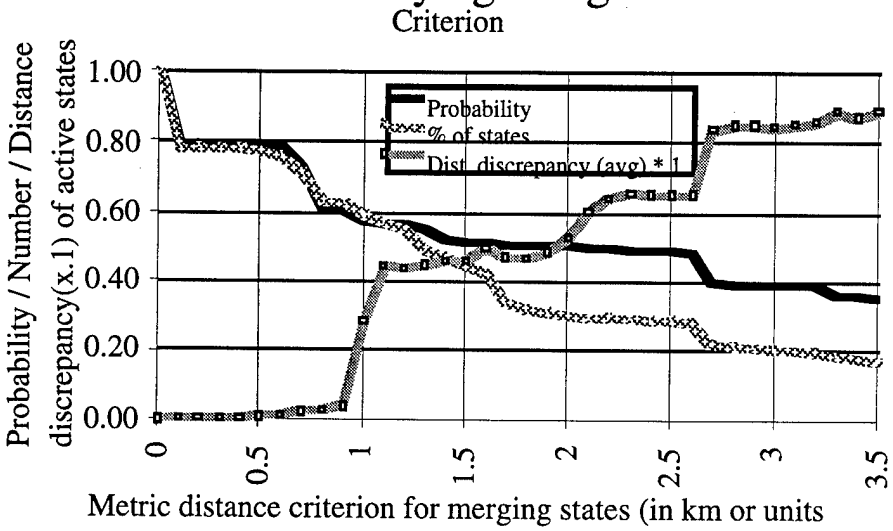
Distance Surrogate Metric

An operationally useful surrogate for distance is needed. For the case below, a vector difference is taken of the sums of the various metric variables (e.g. Blue average X, Y,speed, force,...) in each state. The sume of vector component differences is the aggregate metric used for estimating actual distance, which is the sum of all state vaiable differences.



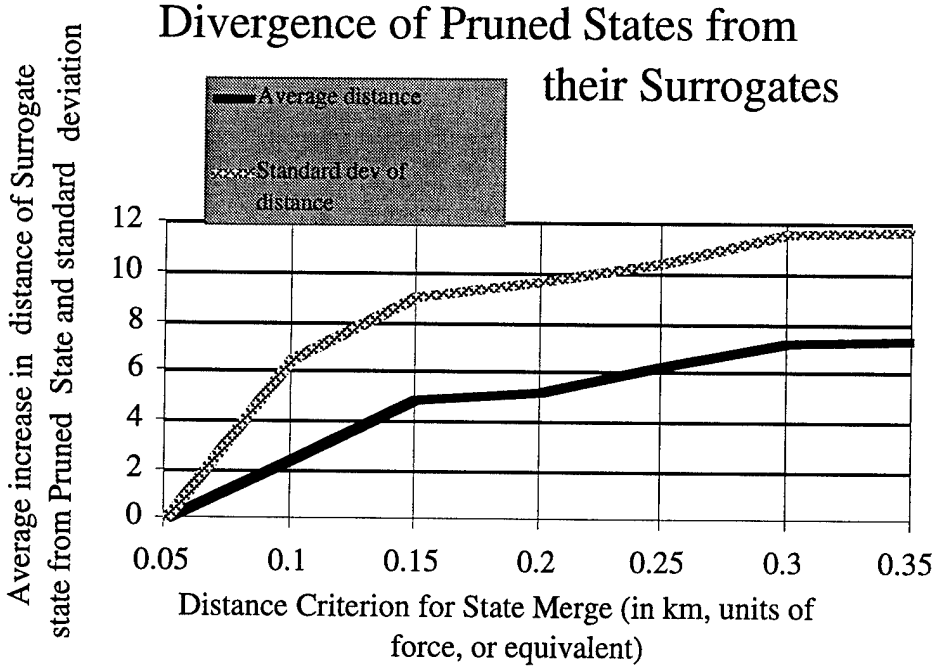
Here we see the actual distances between state pairs compared to the distance metric results. This is quite a bit better than earlier metrics; a scalar metric was almost worthless. The metric used here has about 16 component indicators including aggregate Blue and Red strength, dispersion, mean X and Y location, velocity, and "average" over operational posture.

Effects of Varying Merge Distance



When the value of the distance metric threshold is varied for merging states, we get changes in coverage, and in the number of states kept. It is interesting that even very small distances allow significant reductions in the number of states, with almost no difference in the nature of the outcome space; the average discrepancy between the pruned states and their representatives is small. Many of the merged states are probably similar, but result from a unit taking one path or another and ending up in pretty nearly the same situation. There is a significant jump at "1" for the criterion for reasons that are not now understood.

Very likely this profile would vary with scenario. Toward the right, the coverage statistic does better than the number of states statistic, which is encouraging. Also, there is no very abrupt unbounded rise in the discrepancy statistic, which sufggests that this technique should be useful for pruning large numbers of trajectories where necessary. The figure below indicates that most pruned trajectories styed close to their surrogates, but the very high standard deviation (and a more detailed examination of data) shows that in some cases trajectories deviate greatly from their surrogates. A better metric may prove to discriminate against such cases better than the one now used.

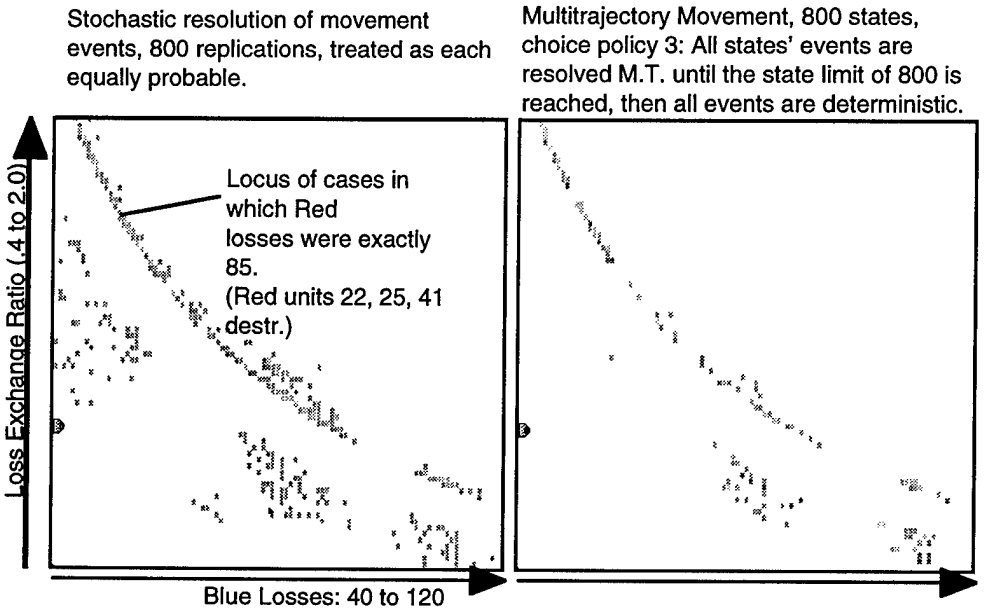


Unfortunately, the version of the simulation that included the state merge capability would not run for the larger "mid-sized" scenario, so we were unable to collect this kind of data for the larger scenario context.

(See following figure) The large scenario runs were used to generate plots that give, in effect, an estimated probability density function over some space defined by two Measures of Effectiveness. Note the difference between this and the outcome plots for the 3 sector Lanchester model, which used input parameter variations for the axes, and could only portray outcome with respect to variation of two parameters. In this plot, in contrast, we reflect the variations due to variability of all events which the analyst has chosen for multitrajectory treatment (In this case, all of the movement choice events).

Note that variations in initial parameters can be thought of as a special case of and event, in which the outcome is a particular value for the parameter. In light of that, the multitrajectory technique could be thought of as an automated method of generating sensitivity analyses, extended to include events that occur dynamically within the simulation as well as initial parameters, and using trajectory probability as a method for accounting in reconning the importance of various possibilities and managing which excursions to examine.

Stochastic and Multitrajectory/Deterministic Policy for Mid sized scenario



Even though Multitrajectory/deterministic gives greater coverage of the Outcome Space, still only a small portion of the outcome space is covered, and the deterministic sampling later in the simulation is not representative.

The results of running the mid-sized scenario revealed the importance of good trajectory choice policy. Here we see both the stochastic and multitrajectory (limited) outcome spaces. In contrast to the smaller scenario, the plot reveals a marked organization, with a locus of points corresponding to battles in which three Red units (those most involved in the battle) were annihilated. The vacant band below results from how unit elimination is accounted. If a unit is below 40%, it is considered eliminated and none of its remaining strength is counted. This gives a quantum effect.

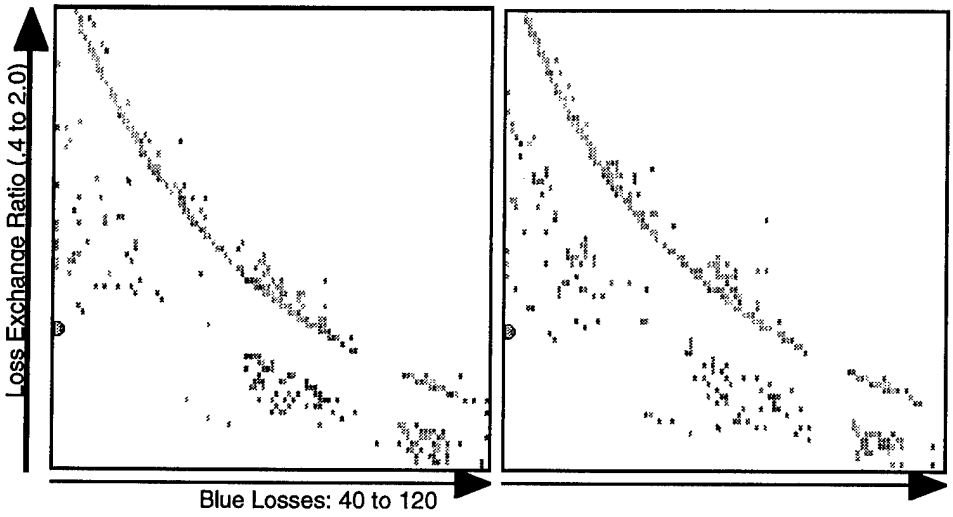
Clearly the Stochastic set tells us more about the outcome space. In the small scenario case, it was possible for the multitrajectory runs to exhaustively cover the outcome space. Here, about three million trajectories would be needed, and the machine used could only handle 800. The choice policy used, "policy 3", used multitrajectory resolution for all events until there were 800 trajectories, then used deterministic resolution. The "budget" of states was exhausted early, before the more interesting variations occurred, so the diversity of outcomes was not uncovered nearly as well as with stochastic simulation. Note that the scattering of outcomes along the left axis is almost completely missed in the (partially) multitrajectory case. However, note that the multitrajectory outcomes show more variation in probablility among those outcomes captured.

For this series of runs, the probability, portrayed as shading, is on an absolute scale rather than relative. The "tab" a third of the way up the left axis is an artifact of the display that was captured for this figure, as is the pointer arrow.

Other Multitrajectory Choice Policies for Mid sized scenario

Multitrajectory Movement, 800 states, choice policy 4: All states' events are resolved M.T. until the state limit of 800 is reached; then all events are stochastic.

Multitrajectory Movement, 800 states, choice policy 6: (After 160 states, only states with p>.00013 have events resolved M.T.; other events are stochastic)



Both of these convey more information than the simplest choice policy, with policy 4 on left giving more information on relative probabilities and policy 6 giving more information on structure compared to stochastic case. A choice policy sensitive to battlefield outcome MOE's may have done considerably better.

With other choice policies, we can actually do better than the stochastic case. Indeed, on the left we see the results of simply resolving events stochastically (rather than deterministically) after the state limit is reached. This has the benefit of ensuring the largest variety of trajectories possible up until the state limit is reached; the previous purely stochastic case would have some trajectories uncovered, and others duplicated, at the same point in the simulation run.

By rationing the last 80% of the states to the higher probability states, a somewhat more sophisticated choice policy, we get the plot shown at the right. This gives somewhat greater (though still small) coverage of the outcome space, while still allowing the occasional outlier outcome to arise.

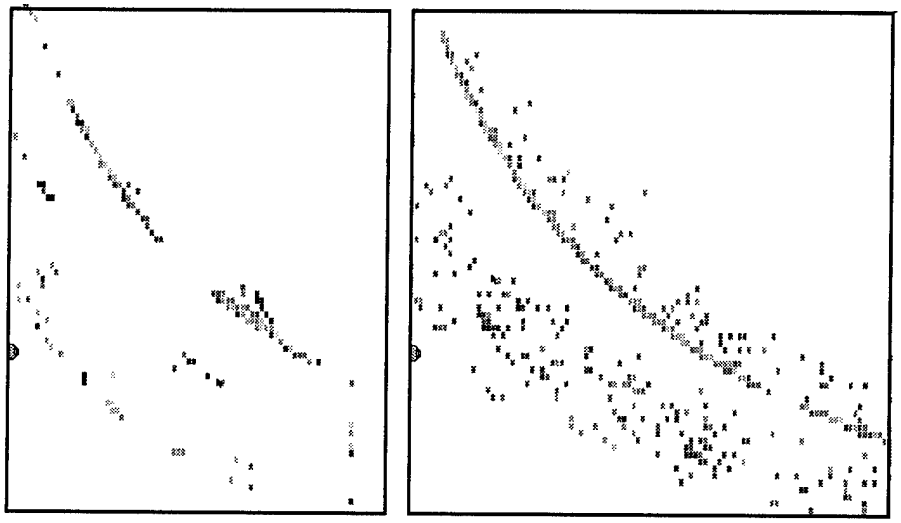
In neither of these cases was there any mechanism to ensure that "interesting" cases were captures. This would be a logical extension to the choice policy library. The Measures of Effectiveness (MOE's) could be utilized as a metric for deciding whether a state should be truncated or not, along with probability and the current number of trajectories.

Note also that choice policy is set at run time, and controls dynamically how events of a given type are resolved.

Other Cases for Mid Sized Scenario

Multitrajectory C2, 800 states, Choice policy 3 (M.T. until 800 states, then deterministic)

Multitrajectory for Movement, Decisionmaking, and Acquisition, 800 states, choice policy 6 for all. (Beyond 160 states, M.T. only if p>.00013)

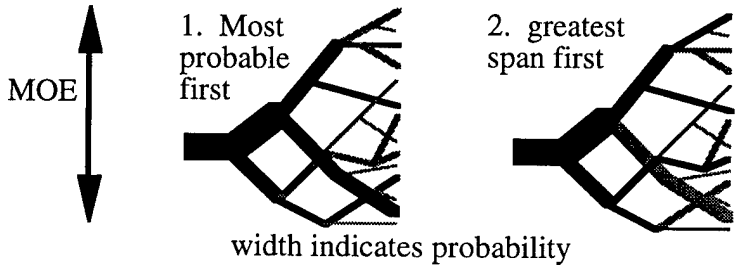


Here we see the results of using multitrajectory resolution of just the C2 events, rather than movement, and the result of using multitrajectory resolution for several events simultaneously.

Analysis Strategies:

1. Manage trajectories to get most probable outcomes first. As more resources are used, less probable outcomes are explored. Stop when the coverage meets your goals.

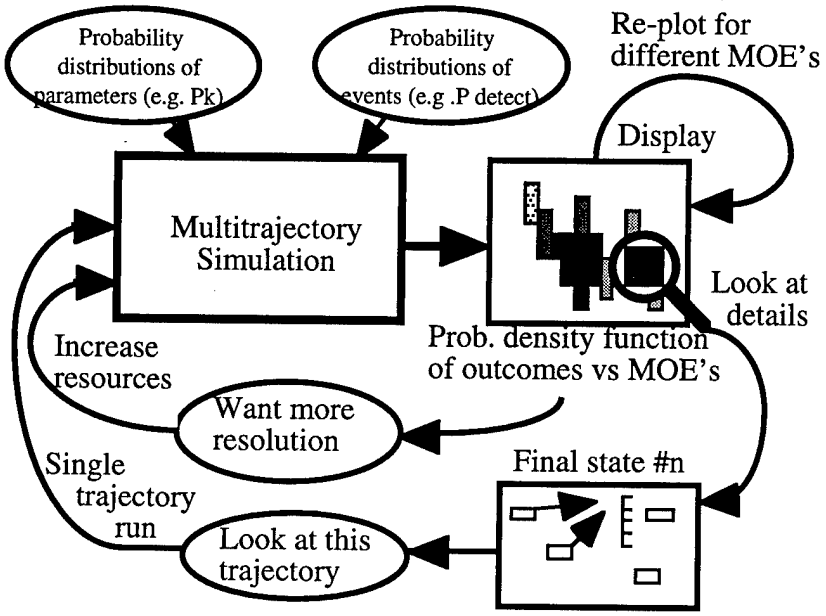
2. Manage trajectories to ensure that the greatest span of outcomes is performed first (evaluated in terms of useful MOE). As more resources are used, trajectories having intermediate MOE values are explored. Stop when coverage meets your goals.



This project could not explore more sophisticated trajectory management strategies such as depth first execution of the more probable or interesting states. We imagine an analyst using multitrajectory simulation in a background mode, with a display portraying the outcome space as illustrated earlier. As more time elapses, more trajectories are brought to completion, and the picture gets added detail. The coverage of the outcome space increases, and the "shape" of the outcome space becomes clearer. When the analyst is satisfied, then the simulation would be halted and the commitment of more resources would cease. It should also be possible to click on one of the cells in the outcome space portrayal, learn which states gave that result, and their probabilities, then replay those trajectories a step at a time to watch the battle unfold. These features have been built into the prototype and some auxilliary software. Each state keeps a record of event resolutions, and the simulation can be operated in a mode where it merely repeats those events according to that script. The "click on" feature, however, is not operational.

In summary, we believe that the multitrajectory approach is part of a suite of tools that will make simulation a more useful tool, with a focus more on the analytic task than the intricacies of simulation internals, while adding little to the difficulty of developing simulations thanks to the class structure that hides the multitrajectory details.

Examples of Analytic Use:



Conclusions

1. It is possible to build a combat simulation capable of multitrajectory simulation, that hides the messy details from the application programmer. (But not as "clean" as desired.)

2. Multitrajectory simulation potentially gives more information to the analyst at less cost, but if resources relative to outcome space size are small, careful trajectory management is needed.

Quantitative results sought still not available; the "eaglet" prototype not fully functional yet.

The results we have achived cannot yet be considered analytic, and have been demonstrated only for a single small scenario. It is clear that a sophisticated choice policy will be needed to get the best results. Thus, the issues of scaling and trajectory management heuristics have been identified as priority issues for further research.

Recommended Further Research

- Investigate scaling issues, with tests involving up to hundreds of units, and larger span of echelons.
- Improve heuristics for Choice Policy:
  When to prune or truncate trajectories as function of MOE's

Exploration of parallel processing of trajectories:
Easier to parallelize than single simulation trajectory

Identification of critical events
With scaling, the focus of what events are critical may shift.

Exploration of more efficient multiple state representations:
Shared objects for more trajectories per unit of computation)

Theoretical basis for understanding multitrajectory simulation
(Perhaps from information theory or other math basis)

Large scale implementation in an existing combat model
(beyond the scope of what Wilkes can do)