MR-388-AF/A

RAND

The Theater-Level Campaign Model

A Research Prototype for a New Generation of Combat Analysis Model

Richard J. Hillestad, Louis Moore

Prepared for the
United States Air Force
United States Army

Project AIR FORCE
Arroyo Center

Preface

RAND has developed and used models of military combat for several decades. Over the years, model-building capabilities have improved dramatically—in computer capacity and speed, software for simulation, and experience in modeling. The end of the Cold War has focused attention on representing and modeling new and different challenges for U.S. military forces, while maintaining a balanced, viable defense force despite downsizing. The *campaign model* has been a key tool in such joint-force structure analysis, strategy assessment, and operational planning. This report describes our research into improving the ability of the campaign model to represent some of the important new characteristics of combat and using new software capabilities to achieve the flexibility required of such models in a highly uncertain threat environment.

We designed and coded the Theater-Level Campaign (TLC) model for RAND's Project AIR FORCE and the Arroyo Center to improve analysis of theater-level joint-force issues in the post–Cold War era. This model introduces advanced features for representing maneuver in "nonlinear" battlefields and structures for better simulation of command, control, and information systems. The model takes advantage of improved graphical user interfaces for setup and output and of state-of-the-art object-oriented simulation software. This report is not meant to be a user's manual for TLC or to promote it specifically. Rather, to help other ongoing efforts move the technology of campaign modeling into the next generation, we describe the lessons we learned in the design process and the specific methods we implemented. Our purpose is to complement the activities of the Air Force, the Army, and others as they examine key joint campaign-modeling issues, capabilities, and requirements for new campaign-model development efforts.

The work described here was conducted in two related projects within two of RAND's federally funded research and development centers (FFRDCs) for national security studies: Project AIR FORCE, sponsored by the United States Air Force, and the Arroyo Center, sponsored by the United States Army. The first project, the Airpower Operations in Joint Theater Campaigns task of the Modeling Improvement Study, was conducted within the Force Modernization and Employment program of Project AIR FORCE and was sponsored by the Director of Modeling, Simulation, and Analysis. The second project, the Nonlinear Combat Modeling Study, was conducted within the Force

Development and Technology Program of the Arroyo Center and was sponsored by the Concepts Analysis Agency.

The report should be of interest to the variety of modelers, analysts, and decisionmakers that use models as part of the campaign-analysis process, especially those involved in defining requirements and constructing new campaign models.

Related Documentation

We have produced five reports and a research brief as part of this work. *Modeling For Campaign Analysis: Lessons For the Next Generation of Models—Executive Summary*, MR-710-AF, provides an overview of important issues and approaches for campaign models designed for analysis. Still to come is the full report, which will address those issues in detail, describe alternative approaches, and discuss how a campaign model should be used in the analysis process. The present report, *The Theater-Level Campaign Model: A Research Prototype for a New Generation of Combat Analysis Model*, MR-388-AF, provides a more in-depth description of the important features we implemented in the Theater-Level Campaign model. The *MAPVIEW User's Guide*, MR-160-AF/A, 1993, describes a graphical user interface developed as part of the theater-level campaign work specifically for the generalized network gameboard that underlies the Theater-Level Campaign model. An earlier workshop report, *New Issues and Tools for Future Military Analysis: A Workshop Summary*, N-3403-DARPA/AF/A, 1992, describes conclusions reached at the start of this project about the need for new models for analysis in the post–Cold War era.

Contents

Figures

Tables

Summary

Motivation for This Research

The bipolar, Cold War era left a legacy that continues to influence both the analysis and the models used in campaign analysis. The models have underlying representational structures that reflect the assessment of defense options of NATO forces facing a Warsaw Pact in Europe. These structures limit the ability to assess new C^4I options, small units, maneuver warfare, new operational concepts, and operations with severe constraints in terms of casualties, rules of engagement, and other factors. In general, the legacy models and Cold War–based defense paradigms limit the ability of analysts to investigate (and conceive) new options for the new situations facing the United States.

Many analysts and decisionmakers argue that an order-of-magnitude leap forward in military modeling—particularly campaign modeling—is essential to improve the quality of analyses, training, acquisition, test and evaluation, and innovative thinking.[1] This research has been a step toward ensuring that the next-generation campaign models will not be mere rewrites of tools we currently use. We investigated and implemented alternatives to four aspects of modeling we think are essential to improving theater-level campaign analysis in the future: (1) how to create more-flexible structures to simulate the wide range of future scenarios and their associated uncertainties, (2) how to link to more-detailed models in an analytically valid way (which we will discuss under the topic of cross-resolution modeling), (3) how to represent ground forces maneuvering at the theater campaign level, and (4) how to better represent adaptive behavior and aspects of command and control in this type of model.

This research, while not resolving all of these issues definitively, has actually implemented specific promising approaches. We built the Theater-Level Campaign (TLC) model to improve our own capability to perform analysis in the post–Cold War era. In many cases, we tried methods and then, finding they were not promising, removed that code and started over with better alternatives. An important lesson was that the development of a truly new campaign model was not just "cutting cookies." We often had to implement our ideas in more

[1]See, for example, Department of the Air Force (1995). Also see the 1995 memo from former Deputy Secretary of Defense John Deutch, which we quote in the introduction of this report.

than one way to determine whether they would work. The various sections of the report describe the results associated with each aspect of our development, describe the implementation we arrived at, and conclude with more general observations and recommendations for the future.

Observations and Conclusions

Flexible Structures

We describe a "generalized network" structure developed within TLC that removes most of the constraints of the more-restrictive piston and regular grid structures of the legacy models, while retaining most of the efficiencies of the more-restrictive structures. This type of game-board approach is considerably more flexible for representing new scenarios and operational concepts.

We have also compared and contrasted various other aspects of the underlying structure of campaign simulations. These include methods to advance time (discrete time step and event step), deterministic versus stochastic representations, object-oriented and hierarchical structures, alternative approaches to allowing human interaction with the model, and approaches to distributed and concurrent processing with multiple processors. Each of these is described and compared in terms of flexibility and importance to campaign analysis based on our experience in implementing a game-board and TLC.

Varying the Level of Resolution and Cross-Resolution Modeling

Variable-resolution modeling refers to the construction of a single model within which analysts can readily change the level of scope or detail at which the phenomena under study are treated. *Cross-resolution modeling* is the linking of existing models with different resolutions.

In TLC, we implemented structures that could be used to represent theater campaigns at various levels of resolution. The structures permit one to select the level of resolution for a particular scenario and problem at the beginning, then to set up and run the model at that level of resolution. We used these structures to represent simulation objects and processes at multiple, selectable levels of resolution. The report describes some of the pros and cons of taking this approach.

A key aspect of campaign models is that much of the system and force effectiveness data must come from other, higher-resolution models. Values for surface-to-air, air-to-surface, air-to-air, and ground force–ground force effectiveness must usually be generated by more-detailed models run for a selected set of input cases. This linkage to higher-resolution models, or *cross-resolution modeling,* is important for conducting high-quality analyses at the theater and operational levels (see Davis and Blumenthal, 1991). We found that, in general, most current approaches to aggregation, while appearing to be logical, have no good scientific or mathematical basis and cannot be expected to provide consistency across the different levels of aggregation, except within very loose bounds.[2] We also found that the strict dependency on other models increases the cost and difficulty of analysis using the theater-level model and that community sharing of data and testing of approaches may increase the practicality and quality of model cross connection. There is a strong need to perform more testing with various approaches to model cross connection.

Modeling Maneuver at the Campaign Level

During the Cold War, campaign models mostly neglected the maneuver aspects of warfare and performed dynamic balance assessments between opposing forces lined up in the pistons of the NATO defensive layer cake, organized around corps boundaries. In the post–Cold War era, the nature of combat is much more uncertain, and the defense doctrine is oriented more toward maneuver. An important goal of the TLC research was to model maneuver in a campaign model. A model capable of examining the current security environment—of analyzing command, control, and intelligence capabilities and considering broad scenario uncertainties in which our forces might be initially outnumbered—must be able to examine the important aspects of maneuver: the competitive "nonlinear" movement of force and fire, the role of information, and the decision cycle. TLC's operational-level C^2 Planner was developed to simulate the execution of an operational maneuver plan by evaluating the likelihood of success of an operational plan allocating reserve ground forces and interdiction assets to maximize the likelihood of achieving the highest-priority objectives, and planning for either the attacker or defender or both. This adaptive approach brings to the operational planning process far more dynamic—and sometimes counterintuitive—allocations of combat and noncombat resources.

Adaptive Resource Allocation

Models that do not react to information and the uncertainty about information by adapting strategies and resource allocations cannot show the value of C^4I assets

[2]These theoretical issues are discussed in Hillestad and Juncosa (1995).

and capabilities. The Sequential Analytic Game Evaluation (SAGE) algorithm of the C^2 Planner within TLC was designed to support the development of adaptive strategies for policy analysis. With the SAGE algorithm, the user specifies a measure of merit (e.g., targets or resources destroyed, final position of the forces, force ratio at the end of the conflict), and the methodology specifies a sequence of strategies for the allocation of the resources of the two opposing sides. SAGE, described in the Appendix B, consists of an algorithm placed within the TLC multiperiod conflict simulation that allows user specification of overall objectives to be minimized (e.g., own attrition) or maximized (e.g., attrition of opposing forces) and engages in an automated search for the "best" strategies to meet the user-specified objectives.

SAGE solutions indicate the strategy that maximizes the marginal payoff of the attack aircraft, whether attacking land forces, air bases, or other targets. SAGE is used in TLC to allocate aircraft, long-range fires, and helicopters to mission types and interacts with the ground force C^2 Planner. It is also used to provide situation-dependent target values for other algorithms. This combination of optimizing algorithms, artificial-intelligence–based scripting rules, and decision tables contributes to treatment of operational strategy as an iterative and adaptive process, while enabling the user to retain control over the objectives and the extent of algorithmic optimization in the execution of an operational plan.

Recommendations

We encourage the campaign modeling community to utilize the lessons learned with this model in terms of approaches to flexibility, model structure, cross-resolution modeling, maneuver representation, and adaptive decision modeling. The separate sections of the report deal specifically with these issues and our approaches. We defined, implemented, and tested important approaches to achieving scenario flexibility and variable resolution in time, space, objects, and processes. Much of this flexibility can be achieved while retaining the efficiencies of less-flexible structures by using generalized networks, event processing, and object-oriented modeling.

The modeling community is only beginning to understand the issues and limitations of the cross connection of models and the approaches to aggregation and disaggregation in achieving consistent results. We know that consistent aggregation can often be done, but a consistent approach requires research with a high-resolution model and is often model and data dependent. We also know that most current approaches are ad hoc and have not been sufficiently tested.

We suggest that other organizations publish their approaches to model cross connection along with testing that has been done to improve the knowledge base in this area.

Covering the variety of situations encountered in a campaign model's algorithms generally requires input of a considerable amount of effectiveness data from high-resolution models to span the range of situations encountered in the campaign model. This process is tedious and time-consuming and somewhat beyond the resource limitations of most campaign-analysis organizations, at least to do it right. Our conclusion is that successful cross-resolution modeling, in which detailed results from high-resolution models feed campaign models, will require data sharing across organizations to provide the variety and depth of cases required. We recommend that data development be organized within the Department of Defense. *We note that this involves the identification of models to be used, cases to be run, and data to be stored; definition of aggregate approaches; contextual description of the data; and development of a process to make the results available to analysis organizations.*

Our research addressed new ways to represent combat phenomena that we think we understand. However, there are many things we do not understand. These cannot be resolved by merely declaring that a new model will "include" those phenomena. *We believe there should be specific research to address how to model poorly understood phenomena, such as information warfare. This research would have a goal of creating consistent models of the phenomena that cross levels of resolution from high to low.*

It is our belief that simulating adaptive resource allocation is a key element to understanding the importance of information and command and control systems. The scripted decisions inherent in many of the legacy campaign models simply do not react to an information war. In Section 6, we give some illustrations as to why adaptive decision processes are important and how they can lead to more robust conclusions in analyses. We developed and tested two approaches to adaptive resource allocation and those are described in this document. We expect that advanced distributed simulation experiments with people in the loop could help further understanding of decisionmaking under uncertainty.

Finally, we note that there are many purposes for modeling and simulation. In some cases, they help structure and think through problems. In other cases, they perform less-expensive testing or training. The use of models for analysis, our primary focus, implies that the models should be transparent, permit sensitivity analysis, be repeatable, and be usable within the time and resource constraints of the analysis requirement. On the other hand, modeling for other purposes may impose requirements for realism, rapid changability, interactivity, etc. It is unlikely that one model will serve all purposes, let alone be useful for the full range of analytic activities. In our discussion of multiple-resolution features, we show that the interaction combinatorics between objects get in the way of having lots of objects at various levels of resolution interacting with all other objects. In fact, inordinate amounts of development time may be required to deal with process interactions that would never occur or are unlikely to occur in the real world. *We recommend that any campaign model be developed with a limited goal for use and range of applications and that multiple models be developed to broaden the range.* As in aircraft design, one model designed for too many purposes is likely not to suit any of them well because of the compromises that must be made. This does not agree with what appears to be the conventional wisdom to consolidate most efforts into a general-purpose model.

An important thread in this report is that many of the model innovations we suggest add a greater burden or reliance on the analyst. To use a model such as TLC—with its flexible structures, variable-resolution features, and adaptive, objective-based resource allocation—the analyst cannot simply turn the crank. The analyst is forced to be intimately involved in the data and structure—a good thing, we think.

Acknowledgments

The authors wish to acknowledge the important and extensive work done by John Owen of the British Defense Operations Analysis Establishment (DOAE) in helping to develop and implement in the TLC model the various approaches described in this report. Eric Larson played a key role in developing data for the model, testing some of its features, and in assisting to develop early documentation. We also want to thank Irene Gordon and Sharon Drummond for their perseverance in helping to prepare and format this document in its many renditions. Finally, we wish to thank Tom Lucas, Bruce Don, Stephen Brady, and Mario Juncosa for their constructive reviews of the report.

Abbreviations

ADS	Advanced distributed simulation
APC	Armored personnel carrier
ATACMS	Army Tactical Missile System
ATCAL	Attrition Calibration [model computation]
ATO	Air Tasking Order
AWACS	Airborne Warning and Control System
BAI	Battlefield air interdiction
BDA	Battle damage assessment
C^3I	Command, control, communications, and intelligence
C^4I	Command, control, communications, computers, and intelligence
CAA	Center for Army Analysis
CACI	[A simulation software company]
CADEM	Calibrated differential equation method [for attrition calculations]
CAP	Combat air patrol
CAS	Close air support
CBS	Corps Battle Simulation
CEM	Concepts Evaluation Model
COFM	Correlation of Forces and Means
COSAGE	[A simulation model used with the ATCAL process]
DARPA	Defense Advanced Research Projects Agency
DIS	Distributed Integrated Simulation
DoD	Department of Defense
Dyna-METRIC	Dynamic-Multi-Echelon Technique for Recoverable Item Control [a logistics model]
EADSIM	Enemy air defense simulation
GUI	Graphical user interface
IDAHEX	[A wargaming model developed by the Institute of Defense Analysis]
JANUS	[A detailed ground combat model]
JICM	Joint Integrated Combat Model
JMEM	Joint Munitions Effectiveness Manual
JROC	Joint Requirements Oversight Council
JSTARS	Joint Surveillance Target Attack Radar System
JWARS	Joint Warfare Simulation

JWCA	Joint Warfighting Capabilities Assessment
MODSIM II	[A simulation programming language]
NTC	National Training Center
P_k	Probability of kill
PPBS	Planning, Programming, and Budgeting System
QJM	Qualified Judgment Model
RJARS	RAND Jamming and Radar Simulation
RSAS	RAND Strategy Assessment System
SAGE	Sequential Analytic Game Evaluation
SAM	Surface-to-air missile
SEAD	Suppression of enemy air defenses
SIMSCRIPT	[A simulation software language from CACI]
STOW	Synthetic Theater of War
TAC BRAWLER	[An air combat simulation]
TAC THUNDER	[An air combat simulation]
TACWAR	[An aggregate theater wargame model]
TLC	Theater-Level Campaign [model]
VUAV	Virtual unmanned aerial vehicle

1. Introduction

The Problem

U.S. military force structure and defense strategy are increasingly determined through the use of computer models.[1] The models are used to game defense strategies, train forces, and analyze equipment for our military, as well as to augment and substitute for live testing of new equipment. New technologies promise to provide even broader use of models in the Department of Defense through advanced graphics, networked simulations, and much faster computation. For example, demonstrations of simulations netted with real exercises fighting against "virtual" and live forces in widely distributed geographic locations have already been performed.[2] The promise of reduced exercise costs, elimination of some high-cost components of weapon tests, and using more-realistic models for analysis continues to drive large investments in modeling and simulation technology and software.[3]

Despite these investments and the impressive technological demonstrations, there are important limitations to the current suite of defense models. The new security environment requires tools that support the examination of first-order issues related to a squadronwide range of nonstandard contingencies and campaigns that are not well represented in the current models. This is because past generations of theater- and operational-level models have been precisely tailored to meet the specific requirements of analyzing only a few "canonical" contingencies—Europe and Korea—where many of the operational and strategic issues were believed to be well understood.

[1]For example, the TACWAR campaign model has been a key analysis tool of the U.S. Joint Staff for evaluating force structure for major regional contingencies. The Air Force uses the TAC THUNDER campaign model to evaluate aircraft, weapons, and operational concepts in joint theater campaigns.

[2]Such demonstrations include the Synthetic Theater of War–Europe (STOW-E) as part of the Atlantic Resolve (formerly Reforger) exercise, the Ballistic Missile Defense Organization Technical Engineering Demonstration, the Defense Advanced Research Projects Agency's (DARPA's) Warbreaker, and the Army's Anti-Armor Advanced Technical Demonstrations.

[3]For a description of the distributed simulation plans, see DIS Steering Committee (1994). It is estimated that DARPA spends $130 million annually on advanced distributed simulation (ADS) activities including Warbreaker and STOW. The Army is estimated to spend $300 million in DIS-related areas.

By contrast, uncertainties abound in the new environment—the theater and terrain; the nature of the forces that will be faced; the coalition, if any, that will be fighting alongside U.S. forces; the influence of new weapon and information technologies; and the operational strategies that might be pursued—all place future defense analysis generally beyond the ability of the last generation of models. The research reported here was thus motivated by developments that suggested that these factors—referred to here as "the elements of nonlinear combat," including a greater reliance on maneuver, the deep battle, "information warfare," and pre-conflict factors—would need to be considered in greater detail in future analyses.[4]

The belief that the fundamental nature of the models in use for defense analysis must change has motivated the Department of Defense to mount an effort to improve its theater-level campaign models. In February 1995, then–Deputy Secretary of Defense John Deutch wrote the Director, Plans, Analysis and Evaluation (PA&E):

> Joint theater models play an important role in assessing the capabilities of our forces and programs to execute our strategy and in measuring the impact of changes in the defense program on warfighting capabilities. As we discussed at the recent review of the mobility requirements update study, many of our analytical efforts have highlighted the limitations of the modeling tools currently available. Those models are grounded in Cold War theory about the use and deployment of forces and the nature of combat operations. They have only limited capability to address key issues from an integrated joint operational perspective. They are also unable to measure adequately the value of the Department's investments in reconnaissance, surveillance, and intelligence and new weapon systems. Nor do they adequately represent the impact of such factors as readiness and training, logistics, or weapons of mass destruction. Moreover, the realism of the basic methodologies that drive the models' results is in need of review. Therefore, as a matter of immediate concern, it is essential to upgrade and refine our current models and to begin development of a new generation of models that the Department will need to address critical joint warfare issues effectively in the future.
>
> Please initiate and lead, with participation from the Assistant Secretary of Defense (Strategy and Requirements), and the Joint Staff, a phased program to upgrade theater models and simulations. In the near-term, the program should improve existing models and in the longer-term develop set of next-generation models for the future. Prepare and coordinate a plan of action, to include funding levels and sources, and brief me on your approach to this high-priority program by the end of February.[5]

[4]See, for instance, Hillestad, Huber, and Weiner (1992).

[5]As a result of the February 1995 Deutch memo to PA&E, a four-pronged initiative began. Prong 1 specifically deals with the near-term enhancement of an existing model (TACWAR). Prongs 2 and 3 focus on future models and modeling environments for the midterm (called the Joint Warfare

The Campaign Model

This report concentrates on the design and use of military campaign models for defense analysis. Models for defense analysis range from engineering models of specific subsystems (a model of an aircraft radar system, for example) to "engagement" models (a surface-to-air system engaging aircraft) to "mission" models (a model of the complete flight of a group of aircraft from takeoff to attack on a target and back again) to the campaign model, which represents a set of missions, operations, or battles in the pursuance of a military campaign objective. Models also exist that have even broader scope, such as the RAND-developed JICM model, which has the capability to simulate global conflict involving more than one theater of war (Bennett, 1994).[6] Campaign models are also sometimes referred to as "theater" models because their focus and scope is usually associated with a full theater of conflict, even though there might be more than one campaign associated with a theater.

Another distinction in models is that between the tactical, operational, and strategic levels of conflict. We will use the term "campaign" model to encompass the operational, campaign, and theater levels of conflict, fully understanding that these represent somewhat different scopes in concepts and analysis. Nevertheless, our research should be applicable for this range of scopes.

Many other model taxonomies reflect aspects of the models other than their scope (Hughes, 1989). These may refer to the type of model in terms of its advance of time, such as time step or event step; in terms of its treatment of random processes, such as stochastic or deterministic; or in terms of its coding style, such as object-oriented or process-oriented code. Furthermore, models may be open or closed, referring to the ability of people to interact with simulated entities during the simulation. The models may be constructed as distributed or single-processor simulations. For the purposes of this exposition, we will sometimes include these various constructs in our discussion but will restrict ourselves to the scope of a campaign model and its use in analysis.

The importance of the campaign model for analyzing defense problems is that, in contrast to models with lesser scope, it is the level at which one considers many of the most important determinants of the outcome of a theater war. The campaign model shows the big picture in terms of the total forces involved, including the joint actions of army, air, and naval forces, as well as the play of

Simulation) and long-term. Prong 4 cuts across all these activities to provide field support of the models.

[6]Daniel B. Fox of RAND has also prepared an introduction to JICM (unpublished).

coalition forces. At the campaign level, one can start to answer the "how much is enough" questions about the military forces, because all forces are represented. Trade-offs between air, ground, and naval forces can be studied at the campaign level. Generally, the measures of outcome of a campaign model are directly related to success in the theater, such as territory lost, achievement of air superiority, and overall attrition in the campaign.

At the campaign level, command and control issues dominate the outcome because we are concerned with how effectively the forces are allocated in pursuance of the campaign objectives. Deployment and sustainment are critical to the success of a campaign, so that the effects of logistics operations are reflected in the outcome measures. This permits the comparison of the effects of improvements in logistics support with improvements in fighting systems. The campaign model is also an important vehicle to study the interaction of strategy, force allocation, and system capabilities. For example, the addition of a more effective surface-to-air capability may permit an allocation of more aircraft to attack enemy ground forces, rather than withholding them for defense. Cumulative effects over time are important in the campaign model, and multiday objectives must be considered, rather than the results of single battles. Finally, the campaign model is the point at which many important scenario factors come into play, such as timing of the deployment of forces, availability of ports and air bases, nearness to ocean access, and strategies of the enemy.

In contrast, the "mission" model may focus more on the timing and details of a single mission, and most scenario, strategy, support, and overall force capabilities are either exogenous or not important. The engagement model further narrows the scope to consider only the starting and ending conditions of a single engagement; the assumptions that led to those conditions may be consistent with the broader scenario, but this is not that important. The engineering or system model cannot by its nature deal with questions of strategy, scenario, or overall force structure. The more detailed types of models provide a more thorough representation of phenomena and address how effective systems are within the contexts of specific situations. The more aggregate, broader-scoped models describe how frequently such conditions occur.

For the foreseeable future, defense analyses will be concerned with two major policy-level issues: force structure—how many divisions, wings, and carrier battle groups (and with what capabilities) there are—and security strategy. Decisions about these will be supported by theater- and operational-level analyses looking at combined-arms operations and will involve cross-service trade-offs and trade-offs between combat forces, on the one hand, and command, control, communications, and intelligence (C^3I) on the other. With potentially new threats and theaters of operation, analyses will also, however, need to address fundamental questions of strategy and operational art; choices in time, space and function, including resource allocations for employment, deployment, and logistics; joint and coalition employment; and exercise of command and control capabilities. The following subsections describe the features of the current defense analytic environment that motivated development of our work and establishes the design goals for the Theater-Level Campaign (TLC) model. Subsequent sections will describe our research toward these goals.

New Circumstances, New Analytic Demands

The current defense analytic environment is characterized by two major features: uncertainty and the need to consider in campaign analyses a number of elements of combat that have lately assumed greater importance.[7]

Uncertainty

The new security environment is characterized by great unpredictability regarding location and adversary, asymmetric strategies, coalition arrangements, and a host of other issues, not least of which are budgetary constraints on analytic resources. This militates against highly detailed planning against possible threats and places a premium on high-quality, quick-reaction analyses to support fast crisis response, yet demands a great many excursions and sensitivity analyses in campaign analyses.

The New Elements of Combat

Consider the host of combat phenomena most dramatically observed in Operation Desert Shield/Storm, presented in Table 1.1.

These phenomena strongly suggest the requirement for greater attention to the modeling and simulation of new elements of combat (many of which were less important in the stylized analyses of warfare in the central region[8]) and consideration of a wider range of scenarios in future defense analysis and planning. Additionally, certain characteristics of warfare that have in the past received limited attention in models (e.g., so-called "soft" factors, such as morale,

[7]For a more detailed treatment of the changed environment, see Hillestad, Huber, and Weiner (1992); Chu (1991); Hollis (1991); Harrison (1991); and Tragemann (1991).

[8]Examples of issues that have been inadequately addressed in the past abound and include maneuver, encirclement, breakthrough, the influence of logistics on maneuver, and the role of forward observers in counterbattery operations.

Table 1.1

Phenomena Apparent in Operation Desert Storm

Maneuver	Air-to-ground effectiveness
Encirclement and force cutoff	Smart weapons
Maneuver of fire[a] (ATACMS, tactical air, cruise)	Tailored weapons/delivery
Air assault	Carpet bombing
Influence of logistics and reconnaissance on maneuver	Weather/day-night influence
	Dependence on air superiority, target acquisition
Attrition (ground-ground)	Stealth delivery
Night vision importance	
Counterbattery capabilities	Deployment and training
Artillery dependence on "eyes"	Limited early capability
Potential for side dominance	Objectives change with capabilities
Casualty-limiting strategies	Capabilities change with training
Fratricide	
	Air operations
Targets	Separate but not independent of ground operations
Strategic	Sustained
Political	
Military	
	Information war
	Target selection
Disruption of Air Defenses	Deception, concealment of maneuver
Effects of removal of C^3I	Damage assessment
Lethal SEAD and reaction of fire units	
Massing and saturation	Coalition, joint, and combined force operations
Jamming	Joint SEAD
Self-protection and stealth	Marine operations
	Tactical air against artillery
	Special forces designation of targets

NOTE: Maneuver of fire means the reallocation of long-range fires to new targets; this can be done without physically moving those assets.

cohesiveness, the fighting effectiveness of different nations' armies, and leadership) may well dominate military outcomes, or—in the case of coalition operations—may constrain operations and will need to be considered more explicitly in future analyses (Bennett, 1992; Davis and Blumenthal, 1991). Finally, recent developments in weapons and doctrine have begun to reach the limits of the ability of the current generation of models to represent them: maneuver, deep fire assets, precision-guided munitions, stealth employment, the heightened importance of the information war, and joint and coalition operations—in short, the elements of nonlinear combat described in Table 1.2. All these introduce new analytic complexities that are not well represented in today's models.

Table 1.2

Elements of Nonlinear Combat

Forces	Command and control
Joint and coalition	Information warfare
Non-homogeneous	New political, military, and tactical objectives/constraints
Technology substitution	
	Asymmetric strategies
	Strategic targeting
Battlefield	
Low density	Maneuver
Non-contiguous areas of operations	Long-range fires
	Air-mobile forces
Attrition	Shaping the battlefield from the air
Non-lethal warfare	Operations in WMD environment
Selective and constrained	Highly mobile ground units/logistics

NOTE: We are using the phrase "nonlinear combat" to connote a departure not only from the linear "piston" models of the NATO–Warsaw Pact conflict studied for decades but also from many of the other standards assumptions of those models.

Joint Issues

The contraction of the defense budget highlights the importance of credible analyses in the joint arena, requiring not only representation at an appropriate level of detail for tactical air and ground forces but also a capacity for exploring air-ground-naval synergisms in different circumstances to identify more clearly and capitalize on the attendant efficiencies and economies of force arising from joint and combined-arms operations.[9] As discussed above, theater-level modeling enables the analyst to assess the implications of various combinations of air and ground forces and can assist in identifying the implications of various types and mixes of air and ground forces, identifying economies arising from synergisms, and developing appropriate operational strategies for employment to optimize critical effectiveness measures.

Coalition Issues

Recent experience in Operation Desert Storm, Bosnia, and Somalia lead one to believe that, in the future, the United States may often choose to operate in a multinational coalition.[10] The vagaries of the coalitions that may be participating in future U.S. military operations raise a host of issues similar to—and perhaps

[9]Joint planning was heavily emphasized in a recent report by the Commission on Roles and Missions (1995).

[10]Army pamphlet 525-5 (U.S. Army, 1994) states that, in future efforts, the United States will typically be the head of a coalition force.

more difficult to address than—those raised by joint operations: command-and-control arrangements, sharing of intelligence data, divisions of labor for air and ground forces, management of multinational combined-arms operations, and so on. Furthermore, coalitions may be dynamic and change during the course of a conflict.[11]

Deficiencies of Current Models for New Security Issues

The current generation of theater campaign models was developed largely to support analyses of a NATO confrontation with the Warsaw Pact that involved well-understood terrain; a reasonably linear battlefield; known forces, operations, and tactics; and highly predictable coalition arrangements. The models did not adequately handle nonlinear deep penetrations and maneuver, dynamic coalitions, adaptive planning, or many of the other demands of the current environment. For example, the Cold War spawned a set of ground warfare campaign models that represent ground operations as a set of "pistonlike" movements. In the Joint Staff's TACWAR model (U.S. Army, 1994b), the Army's CEM model (U.S. Army, 1987, 1991), and the Air Force's TAC THUNDER model (CACI, n.d.), the ground forces fight linear battles along several parallel avenues of approach.[12] This model structure arose from the layer-cake defense posture of NATO corps arrayed against the Warsaw Pact. These models simulated battles that presumed a dense linear cohesive defense that controlled exposed flanks, fell back in coordination with the forces in other pistons, did not perform flank attacks or other "nonlinear" maneuvers, and fought an enemy also constrained to the same pistons. This structure was probably also motivated by the limitations of the computer systems available to analysts at that time, in contrast to the orders-of-magnitude increase in system performance available now. The piston structure executes very quickly but constrains the representation and analysis of maneuver warfare.

In a similar vein, many of the early representations of theater air operations (TACWAR and Combat IV[13]) use "time stepped" models of air and ground operations with relatively large time steps, a few hours to a day. For air operations, the numbers of sorties that can be generated during each time step are accumulated, and their effects are calculated in a process that does not really

[11]An example of this is the possible coalition changes that would have occurred in the Gulf War if Israel had entered the conflict.

[12]The pistons are independent of each other so that they can represent avenues of approach that are noncontiguous. However, the battles are always one-dimensional up and down the piston so that flanking attacks, large scale encirclements, etc., are not easily represented in the linear structure.

[13]Combat IV model is a model used by the analysis staff at the Air Combat Command.

simulate the individual flights, their command and control, or individual engagements. This type of structure, while efficient and satisfactory for many types of analyses, does not permit many of the details of an integrated air defense and its command and control to be represented in detail. The Air Force's TAC THUNDER model simulates flights explicitly, but the flight paths have been constrained to a grid structure that presumes a battlefield like the NATO–Warsaw Pact central region layout.[14] Aircraft fly behind and perpendicular to the forward line of troops, and then cross it when directly opposite the target. This structure causes some representational problems in the simulation of Operation Desert Storm, a war in which aircraft took off from air bases at all points of the compass around Iraq, including bases in Turkey, the United States, Egypt, and Saudi Arabia and aircraft carriers at various locations.

As noted in the earlier Deutch statement, most of the current generation of models have severe limitations for the study of command and control and information warfare issues. Decision processes are typically represented via nonadaptive scripts of orders or simple code rules. This nonadaptivity does not permit the representation of either side to respond to variations in tactics or equipment of either side, as would normally be expected in warfare. This limits the model's usefulness in the investigation of new force structures or strategies in the new security environment.[15] Of course, the scripts can be adjusted by human input for each variation, but this makes for a cumbersome analytical device. It is often not obvious to the user how to adjust the script to take advantage of the new capabilities or to counter those of the other side.

Often, the existing models assume a nearly perfect and instantaneous information flow on the battlefield, and the simulated forces operate without the delays, deceptions, and uncertainties that are normally inherent in warfare. This creates a dilemma when one attempts to evaluate improvements in the C^4I system. There is nothing to be improved on in the C^4I of the model, and any attempt to model it more realistically has the effect of degrading the results.

Another characteristic of the post–Cold War defense analysis environment is that there is a need to evaluate defense options in scenarios of dramatically varying size. On the one hand, there are the major regional contingencies, such as a new Korean conflict or a new Persian Gulf war. On the other hand are the lesser contingencies represented by Panama, Somalia, Haiti, and Bosnia. This variation

[14]Current plans for TAC THUNDER include the adoption of a more general network structure for flight paths.

[15]For example, a response to higher-than-expected aircraft attrition is to constrain operations by region or altitude, or simply to stop flying until the problem is worked out. Most existing models do not go through this adaptation; rather, they continue the operations in spite of the attrition.

in size and character of conflict requires more flexibility and ability to change the resolution than appears possible with the current set of combat models.

The emphasis on joint planning and analysis as exemplified by the current Joint Requirements Oversight Council (JROC) and Joint Warfighting Capabilities Assessment (JWCA)[16] processes of the DoD, and stressed in the report of the Roles and Missions Commission,[17] also means that the current service orientation of many of the models is problematic. The Concepts Evaluation Model (CEM)[18] model of CAA simulates air operations only as they affect ground forces, not representing such other missions as air base attack, strategic targeting, and air-to-air. The Air Force TAC THUNDER model utilizes a simplified version of the CEM model and its data to represent the ground battle while providing a relatively detailed simulation of air operations. Naval/Marine models tend to emphasize carrier operations without simulating the Air Force or details of the ground operations. For credible analysis in the joint arena, it is desirable to provide a balanced representation of each of the service components.

New Software Opportunities

Software and computer hardware have continued to evolve with dramatic improvements in speed, storage, and supporting tools. The improvements include orders-of-magnitude increases in speed and storage capacity, new simulation languages, object-oriented programming structures, greatly improved computer-generated graphics, and networked systems.

Most of the current suite of theater and campaign models were constructed when the hardware was more constraining and many fewer supporting tools were available. This led to particular structural choices for those models (the piston representation, for example) that were compromises between the needs of the representation and the capabilities of the modeling platform. An important motivation for our work in this report was to investigate some of the advantages that the new software tools and hardware capabilities would provide to a theater and campaign model.

One recommendation coming from a May 1991 workshop held at RAND on Future Military Analysis was that the defense analytic community should

[16]As part of the JROC and JWCA, a planning process has been established for joint force requirements and to help prevent some of the "stovepiping" inherent in the current Planning, Programming and Budgeting System (PPBS) process.

[17]Joint planning was heavily emphasized in a recent report by the Commission on Roles and Missions (1995).

[18]U.S. Army Concepts Analysis Agency (1987, 1991).

enhance its capabilities for quick-reaction analysis, since tighter deadlines for analysis and shrinking analytic resources can be expected to combine to reduce significantly the time available for analysis. The models need a flexible, rapid setup capability involving graphical user interfaces and more flexible, variable-resolution structures to permit quick analysis of new situations and the many uncertain features about such scenarios. Thus, we were interested in defining structures to achieve this flexibility in resolution and representation and to permit rapid problem setup with graphical user interfaces.

TLC Research Goals

The research involved in TLC is thus an outgrowth of RAND's past analytic work and a consequence of a recognition that current analytic needs are unsatisfactorily met by the current generation of theater- and operational-level models. The research also builds upon RAND's past work in combat simulation modeling, which includes:

- TAC SAGE, which provides optimized analytic gaming for air allocation[19]
- APEX,[20] which simulates interdiction by fixed-wing aircraft and surface-to-surface missiles and allocates ground reserves using Correlation of Forces and Means (COFM)
- The RAND Strategy Assessment System (RSAS).[21]

A review of other modeling efforts, including those in Europe, has also taken place to ensure that the most current thinking and the latest tools and approaches are incorporated into the model. Thus, in the course of its design and development, TLC has also incorporated learning from other state-of-the-art modeling efforts outside of RAND. In short, TLC is a research effort—not a large-scale development effort—that is aimed at improving the next generation of combat simulation models. With TLC, the intent has been to do this by enhancing some of the important capabilities of RAND theater models with the latest generation of hardware and software. TLC thus seeks to capitalize on the current generation of high-performance workstations (including high-resolution displays) and on window-based user interfaces, object-oriented programming,

[19]SAGE is described in Appendix B. A subroutine called TASK extended TAC SAGE to generate, task, and evaluate ground-support sorties. For a discussion of TASK, see Parker and Wegner (1989).

[20]APEX allocates fires with the TASK algorithm, which attempts to determine the most valuable ground targets (e.g., bridges) to interdict. For a discussion of APEX, see Hillestad, Weiner, and Warner (1991). COFM's origins may be found in Soviet doctrine.

[21]See, for example, Bennett et al., (1993).

graphics, and distributed databases to create a flexible, intuitive graphics-based user environment. The principal research goals for TLC are summarized as follows:

- Develop methods for achieving flexibility in tailoring the model to new environments and problems and to allow quick setup and operation. This includes flexibility in the representational structure of the models, as well as flexibility achieved through user interfaces.
- Develop model software structures that enhance the ability to represent uncertainties, and perform analysis at different levels of resolution.
- Utilize cross-resolution modeling, in which there are clear links to detailed models to ensure credibility and to allow examination of issues other theater-level models cannot examine.
- Develop structures that can represent elements of "nonlinear combat," particularly ground combat maneuver.
- Implement adaptive resource allocation processes that represent goal-oriented behavior in simulated decision processes and that may better represent the decisionmaking components of command and control.

The Remainder of the Report

The design goals for TLC can thus be seen to emerge directly from the challenges of the new analytic environment and the shortcomings of existing modeling tools. The next five sections detail the research to meet these design goals: Section 2 describes our investigation into network structures that permit the tailoring of the model to new environments and problems and graphical user interfaces to enhance rapid scenario development; Section 3 describes our research into model software structures; Section 4 describes our research into cross-resolution modeling in which explicit linkages to more-detailed models provide the data to drive the attrition processes of the theater-level model; Section 5 describes our approach to simulating ground force maneuvers and operational planning; and Section 6 describes some approaches we tried for adapting resource-allocation processes for representing air and ground force allocations. Section 7 provides our conclusions and recommendations based on this research. Two appendices describe the details of simulation process for the allocation of ground force fires and Air Force assets to missions.

2. Creating a Flexible Model Game Board: The TLC Generalized Network

To better meet current defense analysis needs, a new model must be able to tailor the model to new combat environments and problems. Typically, in the initial stages of an analysis, there is a need to strike a balance between resolution and speed in setup and computation, especially when representing a new geographical area. In later stages, increasingly detailed levels of resolution may be required. To facilitate this balancing, we investigated environments that are designed to support variable-resolution modeling, in which the user can specify the representation at the desired level of resolution during model setup and data development.[1] A few examples of variations in resolution include

- Explicit representation of surface-to-air-missile (SAM) laydowns or user-input surface-to-air attrition
- Representation of terrain in detailed grids or as more aggregate regions
- Choices about the size of ground units to be represented in movement
- Aggregate avenues of movement or multiple detailed paths
- Aircraft generation by region or by specific air base or squadron.

The "game board" of a model defines how spatial objects are represented. A key attribute for a campaign model is that the game board have the flexibility to represent different aggregations or levels of detail for the spatial objects, as well as adapting to new scenarios, concepts of maneuver, and geographical regions. Various types of campaign game boards have evolved historically as needed for combat analysis. Unfortunately, the structures campaign models currently use can be quite restrictive in a number of dimensions, as discussed below.

This section begins by describing the evolution of campaign model game boards over the years, as represented in a number of existing models. It discusses the common features and limitations of these structures and then goes on to describe a generalization of the game board that provides considerably more flexibility in varying the resolution and in the ability to represent new regional scenarios

[1]By "variable-resolution modeling," we refer to what Paul Davis and Reiner Huber have called "internally hierarchical variable resolution," in which one can quickly change the level of resolution at which the model operates. See Davis and Huber (1992).

quickly. We spend some time describing how to represent certain combat phenomena in this generalized structure and also discuss some important implementation considerations.

The Historical Use of Network Structures in Campaign Modeling

The Piston Network Structure

The network structure comes under several guises. The most simple and most highly aggregated is the "piston" game board. This structure developed from the need to analyze the NATO–Warsaw Pact force balance in the European Central Region and the fact that it consisted of relatively dense forces organized in a layer cake of defensive corps of NATO allies. The movement of attack and defense forces was assumed to be East and West within these corps areas, and maneuver would consist of reinforcing the various corps from the rear. The model representation consisted of parallel bands called pistons marked off at various intervals into regions called cells. Entities on this structure move and interact back and forth, but not sideways—thus the name. The network structure of this game board can be seen by placing nodes in the centers of the cell boundaries and connecting adjacent cells sequentially with arcs in the direction of the piston. Little, if any, interaction occurs between entities on different pistons. The TACWAR (U.S. Army, 1994a) and CEM (U.S. Army, 1987, 1991) models use such a game board. Figure 2.1 illustrates a piston game board with the overlaid network.

In the piston structure, terrain is represented by the cells of the piston. The boundaries between pistons may be used to represent linear objects, such as rivers. The terrain is assumed to be uniform across a cell. Movement, terrain resolution, and object representation are tightly bound in the piston model by the size of the cells and the corresponding network areas. Objects must not be too large for the cells and also not so small that the aggregate terrain and movement network do not adequately deal with their spatial distribution. Movement is, of course, highly restricted to follow the piston aggregation. An example of a TACWAR piston structure is illustrated in Figure 2.2.

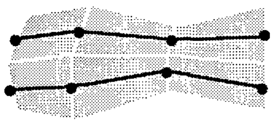


Figure 2.1—Piston Game Board

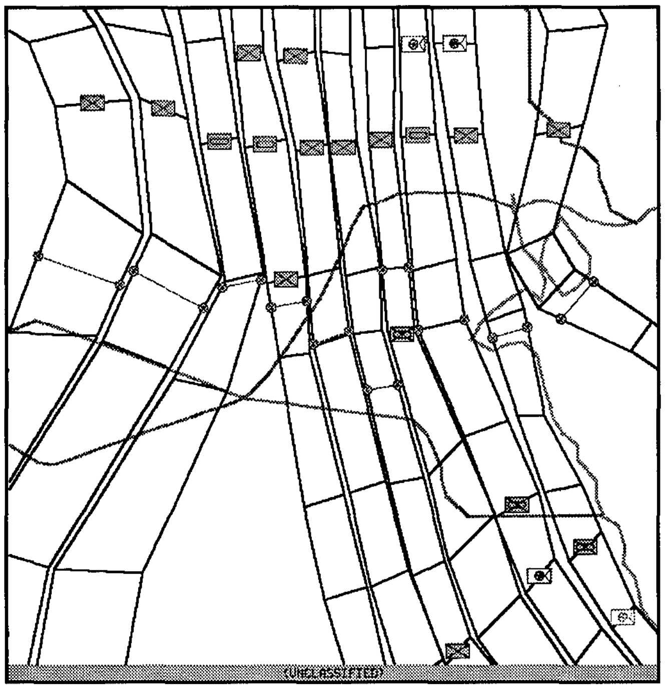


Figure 2.2—TACWAR Game Board

Regular Network Structures

Interactions in four directions may be represented on a square grid structure. On this game board, the area over which entities move is tiled with uniform squares. Placing nodes in the center of each square and the centers of the sides of each square and then connecting adjacent nodes with arcs reveals the underlying network structure. This form is currently used in the TAC THUNDER model for aircraft flights.[2] This type of network is shown in Figure 2.3. This structure, also, was motivated by the NATO–Warsaw Pact layout of forces. It was assumed that an adequate representation of flight paths could be given by aircraft flying parallel to the forward edge of the battle area until they were directly opposite the target and then flying directly east to west across enemy lines to attack the target. This representation caused some difficulties when TAC THUNDER was used to simulate Operation Desert Storm because, in that conflict, aircraft took off from all directions around Iraq as they were launched from aircraft carriers in the Red Sea and the Persian Gulf, as well as air bases in Turkey, the Continental United States, and various Southwest Asia countries.

[2]Current plans for TAC THUNDER are to develop a more flexible network structure.

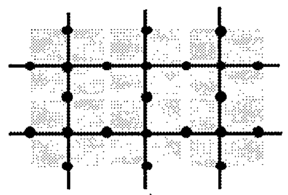


Figure 2.3—Square Grid Game Board

A hybrid between the square grid and piston game structures allows interaction between pistons by distorting the shape of the squares of a square grid so as to cover the cells of a piston game board. This was done in the RAND Strategy Assessment System (RSAS) model,[3] but its successor, the Joint Integrated Combat Model (JICM) (Bennett, 1994)[4] utilizes a more flexible network structure.

The hexagonal game board, another regular network structure, allows for movement in six directions by tiling hexagons over the movement area. The network is formed as in the square grid game, but each node now has six adjacent neighbors. See Figure 2.4. This structure is used in many popular board games and in the IDAHEX model (Olsen, Candan, and deJijs, 1985).

The square grid and hexagonal network game boards fall into the class of regular network structures. In these, each region—square grid or hexagon—is of the same size. The area inside each region is assumed to have homogenous attributes. This simplifies the storage and processing of information by entities on these game boards, because regions may be indexed by vectors of integers and the data for the regions may be stored in arrays for quick access. In addition, since each region is of the same size, distance calculations are replaced by table lookup values. Movement within a region is governed by a uniform set of parameters and is more easily predictable than on an unconstrained game board. Interactions between objects, such as detection, also become easier to represent

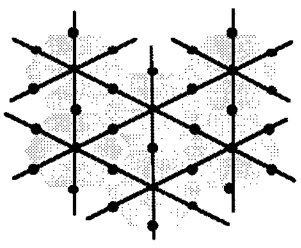


Figure 2.4—Hexagonal Game Board

[3]Bennett, Bruce W., et al., *RSAS 4.6 Summary*, Santa Monica, CA: RAND, N-3534-NA, 1992.
[4]See Daniel B. Fox of RAND's unpublished introduction to JICM.

and predict. Searches may be localized to consider only those entities that are known to be in the same or adjacent regions.

But this simplification into a regular network comes at a price. Information may be duplicated many times for regions representing large, essentially uniform, areas. This may cause needless storage and reprocessing of the information during the simulation. On the other hand, very small areas of uniquely important information may be overly simplified or ignored. In general, the use of a single structure with only one size of region may be inappropriate for all the entities in the simulation. For example, the square grid of the terrain board in JANUS is too small and at too high a resolution to be efficient in representing most of the operations of fixed-wing aircraft. To overcome this problem, aircraft are simulated on another structure by another model and only made visible to JANUS when required, say, to represent air-to-ground attacks (Marti, Kantar, Sollfrey, and Brendley, 1994). In general, using a single, regular network structure as the basis for a model requires that movement, detection, terrain, and unit size all be intimately tied to the size of the cells of the regular network. This can severely constrain the flexibility of representation.

The Distributed Integrated Simulation (DIS), which is composed of multiple models, uses regular network structures and regions of differing sizes in these models, and this causes some interesting problems (DIS Steering Committee, 1994). Ground entities and fixed-wing aircraft both use polygons for terrain, but the polygons of the fixed-wing aircraft simulation are usually much larger than those used for ground-entity simulation. Thus, one of the classic problems resulting from the collision of two levels of aggregation occurs. When a fixed-wing aircraft attacks a ground entity represented on the aircraft game board, the elevation for the ground entity differs from the elevation of that entity on the ground game board. Sometimes this difference is so great that it significantly alters the effects of the attack. The terrain effects can be so great that they alter combat to the extent that tanks appear to fly or to be underground and thus invulnerable to attack. The lesson is obvious. If differing region sizes are to be used, great care must be taken to represent interactions between entities adequately on each. Eventually, as other regular structures with differing region sizes are added to represent additional types of entities with appropriate resolution, much of the value of the regular network board structure is lost in the extra effort required to coordinate the interactions of entities on each.

The TLC Generalized Network

The "generalized network" structure developed during our research with the TLC model eliminates the need for equal-size regions and regularity, yet maintains the efficiency of the network structure. The generalized network is an extension of the regular network but reduces the coordination problems of multiple regular game boards with different-sized regions. As an extension of the regular network game boards, it still possesses much of the efficiency of those game boards but with added flexibility. The price for this added flexibility is the requirement for more analyst involvement in the network development and the necessity of a graphical user interface (GUI) to create the networks.

The generalized network is a combination of "free-form" or irregular networks with free-form "regions." There may be multiple, disconnected networks and many different regions. The networks provide the places that objects might reside and the paths they might take. Each network consists of a set of nodes and, usually, some arcs. There may be more than one arc between the same pair of nodes, and these may represent alternative paths between the nodes. An arcless network might be the collection of nodes representing air bases, strategic targets, etc. The regions are used to represent area objects, such as terrain, country borders, urban areas, coverage of sensors, and lethal areas of SAMs. Regions may also be used to represent regions of command, such as air command and control sectors. In some cases, the regions may overlap; for example, those depicting detection regions may overlap those depicting terrain regions. In other cases, such as when there must be a clear division of command and control, the regions may be contiguous but not overlapping. Figure 2.5 illustrates these various network objects. Regions might be shaped as circles, lines, and polygons. For example, we utilized circular regions to describe lethal radii of surface-to-air systems; lines to define rivers; and polygons to describe weather, command and control, and other terrain regions.

One further component of the generalized network is the "grid." Grids serve the purpose of localizing detections between objects. The grid divides the network and/or the regions into segments or subregions. One form of the grid, called segmenting, divides a set of network arcs into subarcs of equal length. This is illustrated in Figure 2.6. This form of gridding can be used for simple detection and engagement decisions. Objects within the same subarc or within some specific number of subarcs might be considered detected, within fighting range, etc. A rectangular grid that divides a region into square subelements can also be used for detection and engagement. Objects within a certain number of squares or the same square grid element can be assumed to be candidates for detection or engagement, for example. Figure 2.7 illustrates this localization of the detection

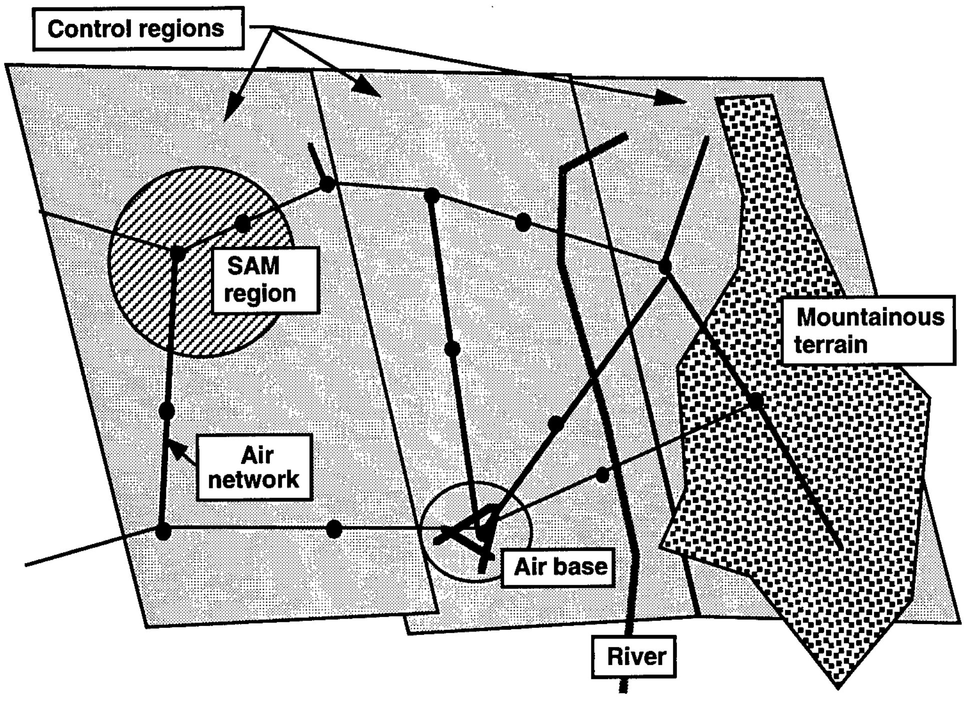


Figure 2.5—Objects in a Generalized Network

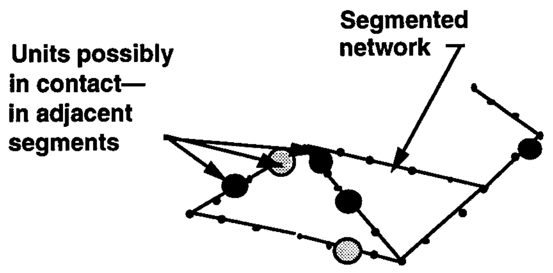


Figure 2.6—Segmented Network

problem. The figure also shows that the use of grids permits objects on different networks to interact. In general, multiple grids are used to define different types of detections or to define the interactions of objects of different types. For example, a space-based detector might use a grid with relatively large grid elements, while the interaction of ground units might require a grid with much smaller elements.

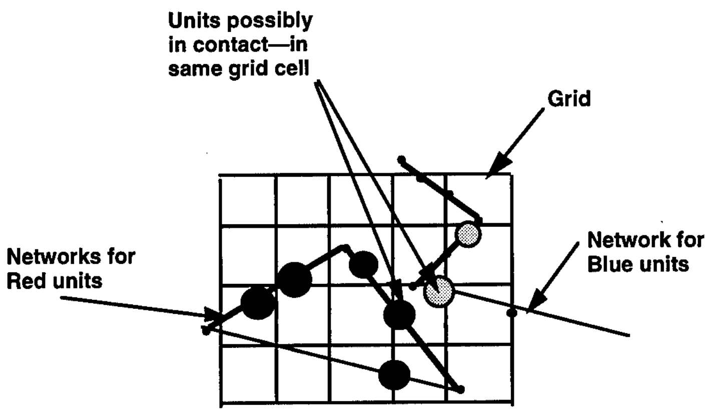


Figure 2.7—The Grid Object in a Generalized Network

Each object—arc, node, region, or grid—has a set of attributes describing it. For example, a terrain region might describe the type of terrain in terms of its intervisibility, foliage, traversability, elevation, etc. A weather region might be described in terms of a name pointing to the weather generator for that particular region. A network of nodes and arcs might describe a logistics network for a side, or it might define the movement options for maneuver forces.

The efficiency of this structure comes from the significant amount of preprocessing that can be performed. The preprocessing determines intersections of networks, grids, and regions. For example, the intersection of a network with a grid creates nodes at each grid line on each arc of a network crossing the grid line. A crossing of a network arc into a new terrain region is also marked by a new node at the point of the crossing through preprocessing. This ultimately facilitates the processing of events in the model. These new nodes, created through preprocessing, might be called "event nodes"; in the model, they trigger events to occur. For example, as an aircraft flight moves along a path of the network and comes to an event node marking the entry into a detection region, the detection event routine would be called to predict the time of detection of the flight. This preprocessing need be done only once for multiple Monte Carlo trials or small scenario variations.

Each arc is assigned the information or events from the regions it lies in. A node that lies within or at the boundary of a region gets the information or event from that region. Figure 2.8 shows an example of such a network crossing two regions.

Each type of entity exists on a network appropriate to its own level of resolution and command and control structure. Each generalized network is governed by

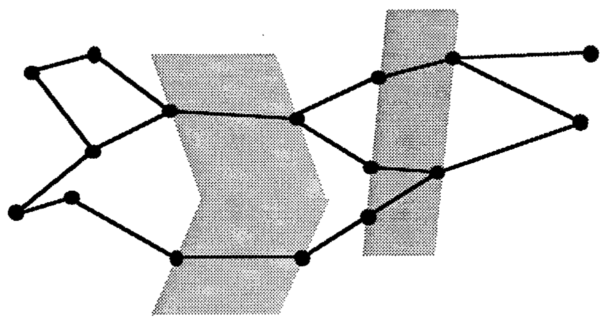


Figure 2.8—Generalized Network Game Board

its own information and event regions or lines. In addition, networks can share some of these regions or lines, to represent the interactions of entities between two networks. The addition of a new network or the modification of an existing network is facilitated by this structure because changes only affect other networks through these shared interfaces.

The resulting network will be flexible and detailed enough to capture the significant information and events required to represent the area of interest, but no more so. Additional regions or areas may need to be added later on, but this can be done rather easily. The price to be paid for this benefit is the increased effort required to define the area, to identify the significant regions therein, and to construct the network. The development of a generalized network is facilitated by a GUI for the interactive presentation and modification of the data of the network. In fact, the productive use of a generalized network probably requires a GUI. We note that there are probably positive benefits in the heavier involvement of the analyst in the simulation structure in terms of increased understanding of the simulation and the causes and effects of events.

Preprocessing a Generalized Network with a GUI

Figure 2.9 depicts the layout of an initial ground network of nodes and arcs, as well as rivers, boundaries, and terrain and detection regions for a Southwest Asia scenario using a GUI tool called MapView (McDonough, Bailey, and Koehler, 1993).

MapView operates on the user-designed networks and regions to create a network with new nodes and arcs at the intersection of every event and information region. This GUI permits a background map to be imported from various sources, satellite imagery, World Data Bank (U.S. Department of

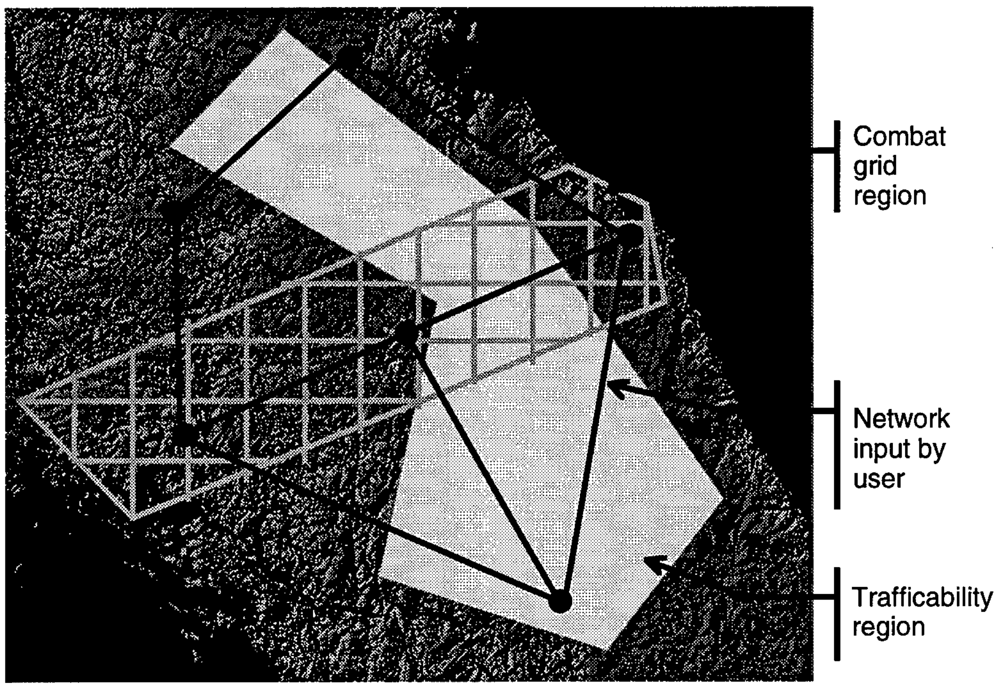


Figure 2.9—SWA Network and Regions

Commerce, 1977), Defense Mapping Agency maps, and scanned maps. The user then draws the networks, regions, and grids of interest on this map background.

The background exists for context only and never enters the model. Figure 2.10 shows the actual network derived from the objects in Figure 2.9. Notice that nodes have been added to the networks where the network intersected with the information or event regions.

Arcs and nodes in overlapping regions are assigned the information from each region unless logic is provided during preprocessing to refine the data. For example, if a water region overlaps a mountainous region, the overlapping terrain might be called water or mountain or both, a mountain lake. In MapView, a special construct is used to partition an area into nonoverlapping regions and to resolve overlapping or conflicting data.

The exact meaning of each region is specified in the data and can thus be changed by manipulating those data without requiring a change to the structure of the network. Thus, very detailed and very aggregate terrain representations may exist for the same network as two sets of data, not as two separate network representations.

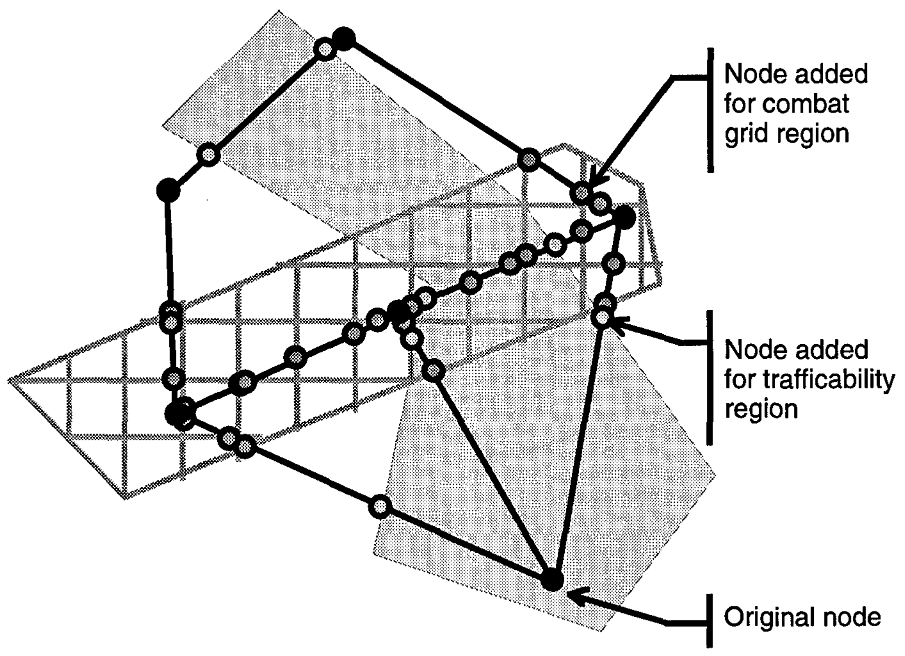

Node added for combat grid region
Node added for trafficability region
Original node

Figure 2.10—Generalized Network Arcs and Nodes

Although automated terrain processing can aid in the construction of these terrain regions, the analyst must be engaged in the process to remove ambiguities and to aggregate needlessly detailed representations. In addition, the analyst must assure himself that the network is drawn fairly. That is, paths should not be drawn around disadvantageous movement regions, either purposefully or accidentally, unless such paths are feasible and units would have the information necessary to select those paths. For example, units with good maps and access to the Global Positioning System would be allowed to take paths around swamps, while those without might be allowed to move into them. Aircraft crews with good information about defenses could avoid high-threat areas en route, while those without would not. Suppose, for example, an air network for Iraq is constructed separately from the air defense regions. When both are combined, one may notice that some air defense regions do not cover an arc from the network and thus will not play in the simulation. Or one may see that some arcs pass through air defense regions that could easily be avoided. Figure 2.11 is an example of such a construction.

Notice that this figure immediately highlights the problem. This shows that the use of a GUI greatly aids the visualization of the game board and data entry for the game board.

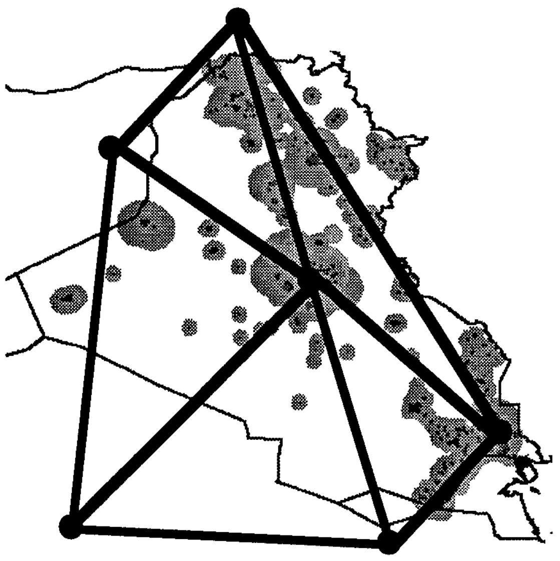


Figure 2.11—An Air Network Overlaid on Air Defense Regions

In general, automated network generation at the campaign level is difficult because of the need to aggregate. Several roads may be designated as a single arc for the move of a division, or routes through a region may be simply defined as a region with a given capacity and trafficability. This aggregation may be better left to the analyst.

Moving Objects on Network Structures

Network structures can be used to define the movement options for entities in the simulation. As such, they constrain directions for movement and permit simplification of movement rules, choices, and decisionmaking. Data associated with the areas and nodes of the network can be used to define capacities, terrain as it interacts with movement, and bottlenecks.

An entity moves on a network structure by dead reckoning along an arc. Two methods are used to depict this movement: "head-to-tail" movement and "point mass." In head-to-tail movement, a moving entity is explicitly represented as a homogeneous object stretching from the location on the network farthest along in its direction of movement, its head, to the least advanced location, its tail. The entity occupies all or part of the nodes and arcs between the head and tail, thus congesting the network. This representation is derived from the way that ground units tend to move along a road network. The bookkeeping necessary to simulate this movement can be quite complicated. Entities tend to move like inch worms along their paths. The problems associated with this method become exacerbated if the entity is allowed to separate or spread out because of enemy attack or movement conditions. For example, if a unit that is spread out on the network is attacked on the flank, there are the problems related to how much of the unit is engaged, how the movement of the head and tail changes, etc. This method is most often used on piston game boards, where entities move up and down the linear structure and do not get attacked on the flank.

With point-mass movement, the entities are disaggregated into one or more subentities. Each subentity moves along the parent entity's path. A subentity occupies the capacity of the nodes and arcs on which it moves for an amount of time dependent upon its speed and size, again congesting the network. But each subentity is represented as a point mass with only one location on the network. Now, if an enemy attacks an opposing entity on its flank, it does not face the entire entity but only the subentities it encounters, as they are distributed on the networks. In addition, units following have the option to traverse alternative paths between adjacent nodes to take advantage of opportunities to reduce the time for the completion of the move or to avoid or attack an enemy. These alternative paths might represent alternative roads or modes of movement. The modeling of point-mass movement is generally simpler than that used in head-to-tail movement—especially when the subentities become widely separated.

In the piston models, objects move up and down the pistons on which they are located and generally do not move between pistons. Each cell is unit-capacity constrained to account for traversability and battlefield width. In addition, because the pistons are the same for each side, the underlying network is also the same for both, requiring each to move along the same routes. An actual attacking force might try to split the opponent by attacking down the seam between two major opposing commands. The piston structure does not allow the explicit representation of this type of attack since one cannot get between two pistons. Detections for close battle occur only near the linear boundary called the forward edge of the battle area between forces on opposing sides.

Movement of entities on regular networks has more flexibility but may be somewhat artificial, moving in a zigzag manner, because of the rectangular or hexagonal structures. This artificiality can cause important distortions in movement distances and interactions with other entities and objects.

Movement of activities on the generalized network structure can be made more realistic and sometimes more efficient in terms of processing. Networks of low resolution can be used to represent paths in areas where few interactions are expected and exact paths are not of interest. High resolution can be provided in other areas of interest by increasing the number of nodes and arcs. Furthermore, multiple networks of different resolution can be used for the movement of different types of entities (air and ground units or combat units and support units, for example).

Representing Detection in Network Structures

The simulation of the detection of objects is a key function of a combat simulation because the detections lead to the command and control decisionmaking and ultimate combat. In a campaign model, the actual sensor-object interactions are typically not simulated, but objects enter detected states under certain triggering events, such as coming within range of an opposing force or entering cells of sensor coverage. Network representations can facilitate this type of representation of detection.

On piston or regular networks, detections are controlled by the cell, square, or hexagon in which the entities are located. Thus, the detection process is tied to the resolution and form of the game board representation. The representation of detection may be simplistic. For example, the location of every entity in the same cell or grid may be known. Or close battle detections may only occur near the forward line of battle area, a boundary between the opposing forces toward which units either advance or retreat. Sophisticated representations of detection may require that each unit continually scan for other units in the same or adjacent cells and grids.

On the generalized network, detections are governed by a subset of the regions, called detection regions, that are overlaid onto the network specifically to represent detection processes or systems. These independent structures are more flexible and adaptive than those used in the regular network structures or pistons. For example, in the piston model, detections for the purpose of close combat occur at the boundary between opposing sides on each piston, the forward line of troops (FLOT). In generalized networks, detections and close combat may occur anywhere. To have this flexibility and efficiency, the analyst may be required to predetermine and lay out detection regions for all sensor-platform-profile combinations for high-resolution representations.

Consider a flight of aircraft on the generalized network structure. As the aircraft moves along its path, it may enter the node at the boundary of a detection region or line. At that point, a process is triggered that computes the time—perhaps stochastic—when the flight will be detected and intercepted by opposing aircraft. The time depends upon the penetrating aircraft and the enemy defenses. Several types of detection processes may be represented this way, each with its own detection region. Several such regions may be encountered along the flight's path. A highly aggregate model will have a few, a detailed model many. Thus, preprocessed data control the detection event execution and level of aggregation, while the same modeling event routine can be used across all cases. Figure 2.12 illustrates this type of detection modeling.

One of the difficult problems associated with campaign simulations when they have many combat entities is that each simulated object must often determine its nearness to all other objects to determine if it is in detection range or within a range at which it must do battle. This can involve many n x n distance calculations when n objects are represented. With a generalized network structure, it is possible to localize the problem so that such range calculations are performed only for objects in the appropriate vicinity. This is done in the following manner.

The analyst decides the area in which detections might occur. This area is partitioned into nonoverlapping detection regions that depend upon the detection process. When an entity enters the node at the boundary of any detection region, it is entered into a list of entities in that region. When it departs the region, it is removed from the list. Since each entity moves along its own network by dead reckoning, one may predict the next entity it will detect or contact. The only entities that need to be considered are those in its own and adjacent detection region lists. The search has been made efficient through

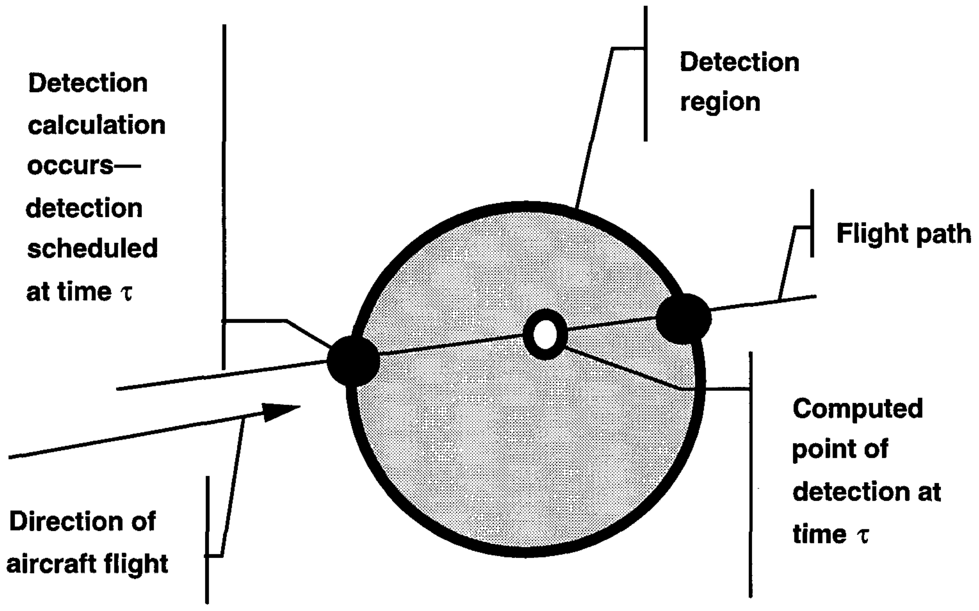

Detection calculation occurs—detection scheduled at time τ
Detection region
Flight path
Computed point of detection at time τ
Direction of aircraft flight

Figure 2.12—Predicting Detection Time in a Detection Region

localization to the detection regions, and the search itself is executed only once for each detection. Frequently, the detection regions are square grid elements, but this is not required. Figure 2.13 illustrates this process.

Note that entities may move on different networks and are not necessarily the same type of object. That is, aircraft may detect ground units using this scheme.

SAM defenses may detect and attack aircraft as well. An interesting example of this type of detection scheme is the observation by overhead and wide-area sensors of ground units, using the Joint Surveillance Target Attack Radar System, or of air units with Airborne Warning and Control System aircraft. As the sensor moves on its network, it monitors one or more adjacent detection regions. Units in those regions become eligible for detection when they enter the region. Not all such units are detected, and the information is not globally available. The sensor must report this information along its communication network for the detections to be further processed. Figure 2.14 illustrates this example.

The exact structure of the detection regions is governed by the detection process itself and should represent the command, control, communication, computers, and intelligence (C^4I) for that process. This process is not hampered by the structure of the movement network or other detection networks. The flexibility and adaptability of the generalized network structure allow additional detail to

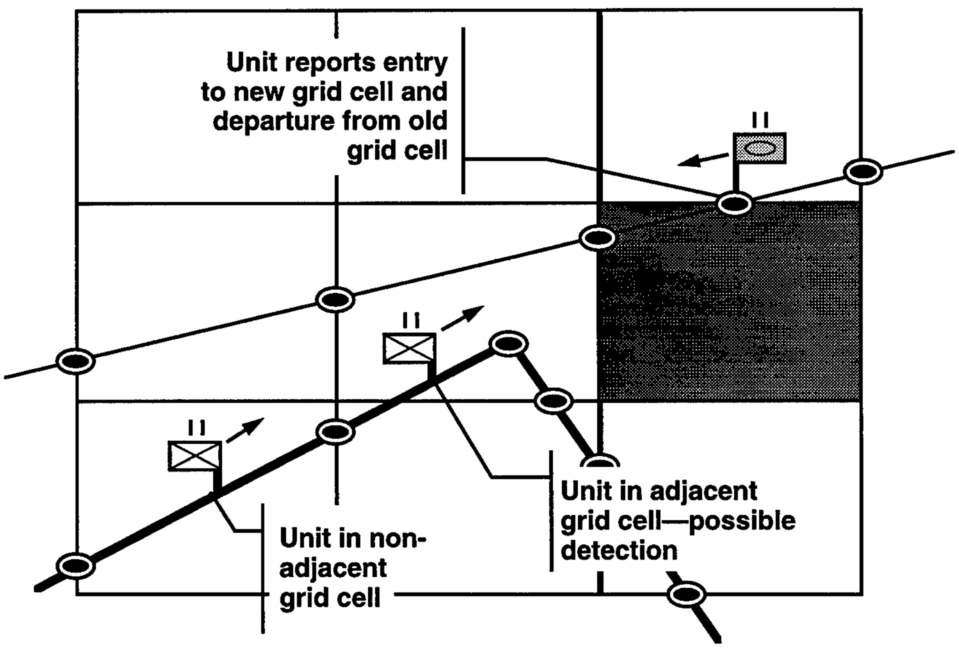

Unit reports entry to new grid cell and departure from old grid cell
Unit in adjacent grid cell—possible detection
Unit in non-adjacent grid cell

Figure 2.13—Use of Grids for Detection

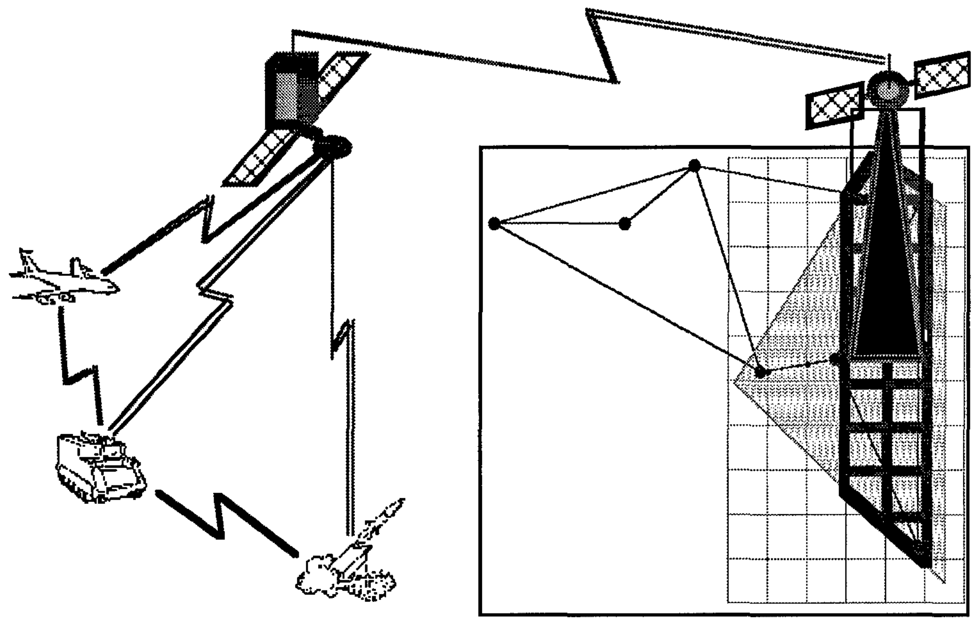


Figure 2.14—Detection by Overhead Sensor

be added or removed from the detection process without disturbing the structure of other networks. Again, this entire process is aided by and probably requires the use of interactive GUIs.

Routing Objects on Networks

One of the key advantages of network structures is the ability to define routes quickly for objects in the campaign model. In piston networks, the routing is simply the movement up and down the arcs defining a piston. In a regular grid structure, the shortest routing between the center of any two grid elements is easily defined. In the generalized network, the routing must be determined algorithmically. Shortest-path algorithms, operating on node arc networks, can execute quickly to determine the fastest route between any two nodes of the network. In fact, most shortest-path algorithms can be used in a preprocessing mode to predefine the shortest routes between any two points in a network. Thus, during the simulation, the route determination is merely one of looking up the path. "Shortest" can be defined in terms of distance, time, and even detection probability.[5] In combat, however, the shortest path may not always be the most

[5]For a route that minimizes detection, if one can define a detection probability on each arc, the minimum probability of detection is on the path that maximizes the product of nondetection on each

desirable. It might be better, for example, to take the path that is the shortest but with the least danger. Algorithms that automatically find such paths can also be defined. In TLC, we defined an algorithm to find the least threatening path for an aircraft flight that was within the range constraint of the aircraft. Thus, with the generalized network, as with the simpler network forms, it was possible to define routing with great efficiency and with attention to avoiding or routing around threats.

Frequently, it is desirable to obtain the routing to an object that is not on the movement network of the object being routed. An aircraft attack of a ground unit is one such instance, because the ground unit may be moving on one network and the aircraft on another. The routing can be determined by associating with any network a set of contiguous regions encompassing the areas to which objects will be routed. The aircraft routing network might then have a set of regions called "flight-path regions." The ground unit movement network would be preprocessed against this set of regions so that, as ground units enter a new flight-path region, an event is triggered that causes the flight-path region to store the ground unit as one of the entities it contains. Also associated with the flight-path region would be a node of the flight path network that represents the point to fly to when going to that region. Then, when the aircraft must attack a ground unit, the location of that ground unit in terms of flight-path region is obtained, the flight-path node of the flight-path region is determined, and then the aircraft is routed from its base to that flight-path node on the flight-path network. Depending on the level of detail and aggregation desired, the flight-path regions can be made quite small and the network detailed, or they can be made large with a more aggregate network. Figure 2.15 illustrates this process.

It is also possible to "fly off" the network to the specific location of the ground unit from the flight-path node by adding a time delay corresponding to a distance calculation from the node to the ground unit. However, when objects move off the network, they no longer have the advantage of the preprocessed network information; the determination of defense interaction, terrain, etc., must be done more dynamically, losing much of the value of the network efficiency. It is probably more desirable to increase the complexity of the network and to keep objects constrained to it.

arc. By using the logarithms of the nondetection probabilities on each arc and maximizing the sum (because the logarithms are negative, this is like minimizing the sum of the negation of the logarithms), it is possible to use a shortest-path algorithm to define the path with the lowest cumulative detection probability.

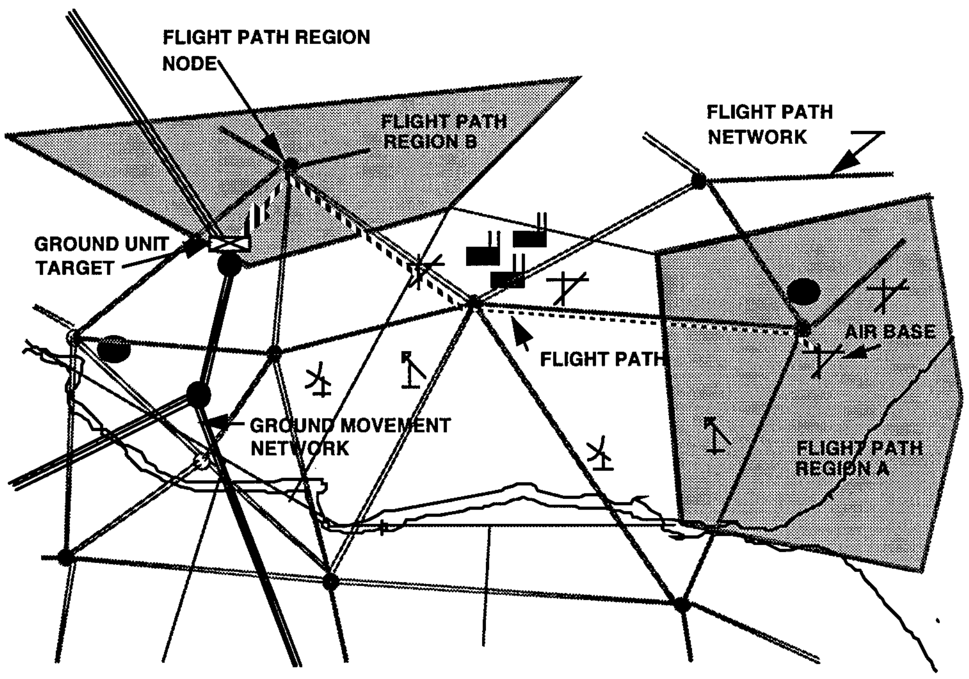


Figure 2.15—Path Determination for Objects on Different Networks

Dynamic Network Objects

Preprocessing is important to attain the efficiency of the generalized network. However, some of the regions that one needs to use in a combat simulation are more dynamic. Regions of surface-to-air coverage may change as defenses move around. Bridges may be destroyed, so that capacity and trafficability change on a route. And as regions of terrain are won or lost, air bases may be closed, sensors may be moved, and the routing of ground units must change. These dynamic changes can be handled in several ways with the generalized network structure. Additional regions can be predefined to take into account such contingencies, and these can be activated as the situation in the simulation changes. For example, several detection zones could be defined as contiguous polygonal regions, and as defenses move, the characteristics of these regions would change to accommodate the new situation. As the characteristics of arcs in the network change, it may also be desirable to recalculate the shortest or least-threatening routings. These approaches permit dynamic changes to the network but retain the advantages of preprocessing.

Some Final Comments About Network Structures

The generalized network structure includes the piston and regular grid structures as possible representations. It achieves the most flexibility, with the exception of a completely unconstrained structure, for the representation of movement, terrain, detection, and C^4I in a campaign model. It is a variable-resolution structure that also permits significant computational efficiency and is highly suited to event process time management. It is made possible by advances in GUIs.

On the other hand, for many analysis problems, the piston or regular network structures are perfectly adequate and carry much less overhead than the generalized networks. They do not require GUIs (as much), do not require preprocessing for efficiency, and do not generally require as much modeling or analytical expertise to implement. The choice depends on the needs of analysis and the desire to have variable resolution and flexibility imbedded in the model.

3. Flexible Software Structures for Campaign Models

The Importance of Model Software Structure

The choice of software structure is one of the most important decisions with respect to campaign-model development and is the basic foundation on which the simulation is built. The underlying structure of a campaign model refers to its treatment of simulated time and events, its representation of spatial objects, its approach to representing command and control, its representation of random events, the type of software construction used, the choice of distributed structures, its interface with operators, and whether or not the model has variable-resolution capabilities. The structure affects the level of resolution possible with the model, the efficiency in processing, the adaptability of the simulation to new problems, the transparency of cause-and-effect relationships, and the ease with which future changes can be made and generally limits the analysis applications of the model. Structural choices must be made early in the design phase of a model and subsequently affect all other aspects of model development.

We have already discussed a model structure for spatial representation in the previous section. This section addresses our investigation of other aspects of structure necessary to achieve flexibility in a campaign model.

It is not so much that any specific structural choices are inherently bad but that their limitations should be understood when the choice is made. Furthermore, some structures are less limiting in terms of campaign-level representation than others. For example, as we have shown, a piston model cannot be easily used to represent complex maneuvers, but a more generalized network structure can represent piston movement as well as nonlinear maneuvers. A Monte Carlo model can simulate both random and deterministic events, but a deterministic model, by its nature, does not simulate random events. In the remaining subsections of this section, we will describe and critique some of the structural choices we have studied within TLC for campaign level simulation.

Components of Structure to Be Discussed

The components of structure of a campaign simulation we will discuss are

1. Simulation time management—whether time is advanced in event steps, in fixed time steps, or continuously
2. Stochastic structure—random or deterministic
3. Software structure—the software paradigm, such as object-oriented coding, structured programming, and process-oriented programming
4. Distribution structure—distributed processes, distributed trials, single processor, parallel processing, etc.
5. Resolution structure—fixed, variable, selectable resolution.

The following subsections discuss each of these structures from the standpoint of a campaign model designed for analysis. In Section 6, we will take up the discussions of the command and control architecture—whether the model is closed, open, or scripted; uses a human in the loop; has semiautomated planning; uses a decision table; etc.

Simulation Time Management

One of the primary considerations of a simulation is the method used to advance time.[1] Three basic methods are used to move through simulated time on digital computers: continuous approximations, time steps, and event steps. Continuous time advance is approximated by the solution to differential equations. In reality, we look only at the solutions to the differential equations at discrete points in time, but the continuous solution is approximated by a series of iterative steps that reduce and control the discretization and rounding error. Most campaign-level models are not represented by a set of differential equations and consequently do not use the continuous approach.[2] The time step has been historically popular because of its simplicity, the ability to create relatively aggregate models, and the degree of control over the processing sequence. The event-step method simulates events when they occur during a simulation and provides an efficient way to advance time in large steps when nothing is

[1]There are, of course, models that do not advance time. Analytic queuing models, some cost models, etc., do not explicitly represent time. However, campaign models are generally simulations that must explicitly advance time.

[2]The exceptions to this are models using the Lanchester differential equations of combat. However, with constant coefficients, these can be solved analytically; in other cases, they have been approximated with the time-step methodology.

happening and to use very small steps when important events occur. The event step method schedules and unschedules events, moving forward in time by stepping to the next earliest event, then processing only that event. This new event may include the scheduling and unscheduling of additional events.

The following subsections describe these alternative approaches to time management in more detail.

Time-Step Method

Suppose we wish to simulate the state of a dynamic system, say the detection of penetrating aircraft, over some time interval. Let S_t denote the state of the system at time t. We are interested in simulating S_t for t between some initial time, t_0, and final time, T. Figure 3.1 presents a hypothetical time line for the system. The times for the discrete events of the system, aircraft detections, are indicated by the set $\{E_i : i = 0, 1, \ldots\}$. The times for a time step of fixed length, $\Delta t = t_1 - t_0$, are indicated by the set $\{t_i : i = 0, 1, \ldots\}$.

In the time-step method, the next time step is determined exogenous to the simulation. Events for a system are not simulated at their exact time of occurrence, but are aggregated to the end (or beginning) of each interval. Consider the simulation of aircraft detections with a fixed time step of $\Delta t = t_1 - t_0$ The initial detection is simulated at time t_0. By t_1, no detections have occurred, so nothing needs to be simulated. During the next interval, two detections occur. Both are simulated at t_2.

From Figure 3.1, one can observe that a simulation using a small time step will tend to have smaller deviations from the actual system than simulations with larger time steps. The choice of a small time step minimizes both the likelihood of multiple system events occurring in the same interval and the difference between the actual time of occurrence of the event and when its occurrence is simulated.

Each state of the simulation depends on its previous state. Thus, deviations occurring at early time steps in the simulation may interfere, constructively or

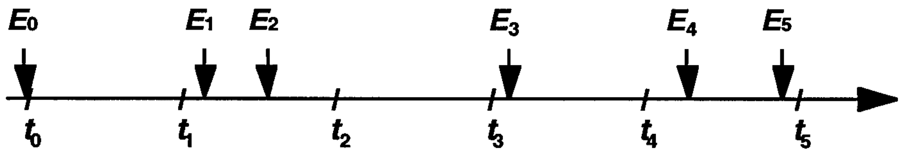


Figure 3.1—System Time Line

negatively, with the events in subsequent time steps of the simulation. This implies that the likelihood of more-significant deviations of the simulation from the real system tends to increase with time.

Using a smaller time step may not be a practical alternative in all situations. Reducing the time step increases the number of time steps that need to be simulated and increases the likelihood that many time-step intervals will include no events. Both effects increase the effort, sometimes needlessly, required to perform the simulation. For some systems, the step size for the simulation may be so intimately related to the intrinsic nature of the system that changing it may not be possible. In addition, smaller time steps may cause numerical round-off instabilities or discretization errors to occur, because of the precision of the representation of various quantities during the execution of the simulation (Press, 1989). Requirements to maintain integrality of systems may also limit the choice of time step. If, for example, the number of kills in an interval is estimated by taking the average firing rate times the probability of kill, but the time step is so small that this computes to a fractional kill, then rounding to an integer number of kills may grossly overestimate or underestimate the correct number over the course of the simulation.

A time-stepped process can be implemented with some additional flexibility. It may be desirable to increase the accuracy of the simulation in some areas but not in others. These processes may be simulated with smaller time steps than the others. For example, the weather-generation process of a simulation could be performed on a daily basis, while the movement process could use an hourly time step.

The important advantages of the time step method are its simplicity and the degree of control one can establish regarding the order of processing. As we will discuss in the next subsection, the order of processing is strictly defined at each time step so, for example, the high level planners develop their plans, the midlevel planners then develop theirs, and finally the lowest-level planners produce their plans, each using information from the other.[3]

The time-stepped approach is also used in highly detailed simulations and games that require very small time steps.[4] In such simulations, it is often easier to check all processes at every step than to attempt to predict the next event. For example, when aircraft fly through a dense radar coverage, it is easier to move the aircraft

[3]In an event-stepped simulation, unless precautions are taken, the processes to be done at the same time may get done in the order that the events appear in the event queue rather than the desired order.

[4]The Army's JANUS model and the Corps Battle Simulation (CBS) model, for example.

a small distance along its flight path and then make detection calculations for each radar system given the exact geometries than it is to predict an event time of the future at which detection would occur. Also, some processes, such as planning, day, and night, have natural time steps.

The primary disadvantages of the time-stepped approach are the potentially large approximation errors in large time steps and the possible inefficiencies when using very small time steps. Furthermore, variable resolution cannot be achieved by merely changing the time-step size without also changing the object representation in the simulation, because time steps that are too small can cause integrality problems, and time-steps that are too large may require too many implicit assumptions about what occurred during the time step.

Event-Step Method

In the event-step method, the length of the next time step is a variable determined within the simulation.[5] When using this method, at any time t, the occurrence time of the next simulated event in the system is predicted. For example, in Figure 3.1, the time for the next detection of an aircraft after time t_0 is E_1. The prediction of the next event is crucial to the event-step method. Suppose that, at time t_0, a detection has occurred. When using the event-step method, the simulation would be required to predict the time of the next detection,[6] in this case E_1. The simulation might use dead reckoning based on the flight paths of the aircraft in the simulation to compute the next detection time and the aircraft to be detected.

At time E_1, a detection would occur unless the paths of the aircraft in the simulation had changed in the meantime. An aircraft changing its path causes the simulation to adjust the time of occurrence of the next detection or perhaps cancel the if a detection will no longer occur. When a detection event does occur, the time of the next detection, E_2, would be predicted, and the process would continue.

The event-step approach to time management provides for somewhat greater efficiency than does the time-step method (for some objects), in that it allows for large advances of time when nothing is happening and very small steps when very much is happening. Modern simulation languages, such as SIMSCRIPT (Kiviat, Villanueva, and Markowitz, 1984), are organized around an event-step

[5]See Fishman (1978), for a complete treatment of this subject.

[6]Or next "possible" detection. This could be done simply by stepping ahead to the next radar sweep.

approach so that the developer is not required to build the necessary library of routines for managing the list of events (called the event queue).

The primary disadvantages of the event-step approach are the need to be able to predict the next event, the difficulty of controlling the order of event processing for simultaneous events, and the skill required to reduce a process to a series of events. We have already briefly discussed the problem of predicting the next event. Sometimes, given a highly complex set of interactions, or an almost continuously variable state, it is difficult to determine when the next event will occur. Given many radars observing many flights, the equations to predict the time of detection may be formidable. It may just be easier to use a small time step to advance the aircraft a short distance, then calculate who might have detected it in that interval. Of course, there is nothing that precludes using the small time step in the event-stepped method as the next event to process, but this does preclude the efficiency argument made earlier.

The problem of the order of processing simultaneous events is a common one. Scheduled events are generally stacked in a queue, and those to be done at the same event time are typically processed in the order they entered the queue. It may be that the planning functions for several different levels of a command hierarchy are simulated to happen simultaneously but that the order of processing is important. The higher-level commands must determine and pass their orders on to the lower levels before the lower levels can properly plan. But if the lower-level planning function were somehow scheduled first, the sequence would not occur properly. This can be controlled in two ways, first by making sure that the processes are scheduled with a small amount of time between them to cause the proper order to occur or, second, by obtaining control of the event queue and the order of event processing for those events scheduled at the same time.

Developing and using an event-stepped simulation requires somewhat more skill than models with fixed time steps. However, the efficiency advantages, the almost infinite ability to vary the time resolution, and the ability to avoid making implicit assumptions about processes occurring during fixed time intervals can make event-stepped time management the time structure of choice. The fact that time steps can be represented as event steps in an event-stepped model (although less efficiently) means that this approach provides a flexible underlying structure for simulating complex systems in which there may be natural time steps but that also have other asynchronous processes.

TLC was implemented as an event-stepped simulation in the MODSIM II language (CACI, 1991). This language provides a library of event management routines so that the user does not need to program his own. Event management is provided in most event simulation languages such as MODSIM and SIMSCRIPT (Kiviat, Villanueva, and Markowitz, 1984).

Simulating Random Processes

Many of the early and current campaign simulations are deterministic models. Because large numbers of events occur during fairly large time steps in these models, mean values of attrition and system states are considered representative and subject to the law of large numbers.[7] These deterministic models run quickly and produce repeatable results given the same starting conditions, making them very attractive to the analysis process. As time steps grow smaller and as more-detailed representations of individual objects become important and feasible, the deterministic approach encounters important limitations. Much about combat at higher resolution appears to be stochastic. At higher resolution, it is much harder to appeal to the law of large numbers because the numbers are not large. The detection of a single flight of aircraft by various radar sensors must be a discrete event that happens at a random point in time depending on the flight path, the sensors, the operators of the sensors, etc.—the aircraft cannot be half-detected. Similarly, a ship cannot be half-killed by a missile given a 50 percent probability of kill.

Consider the following example of the effect of random attrition in a ground-combat engagement. It demonstrates that results for stochastic behavior are very different from those for a deterministic engagement with the same parameters.

A battle is simulated over short time intervals of length Δt with the respective numbers of the Red, R_t, and Blue, B_t, systems surviving the battle at time $t \geq 0$ satisfying the system of difference equations,

$$\begin{aligned} R_{t+\Delta t} - R_t &= -a\, B_t\, e^{\sigma Z_t^B} \Delta t \\ B_{t+\Delta t} - B_t &= -b\, R_t\, e^{\sigma Z_t^R} \Delta t, \end{aligned}$$

where σ is a non-negative constant and Z_t^B and Z_t^R are independent standard normal random variables. The battle terminates when either side suffers 55-percent attrition to its systems; the other side is then declared to be the winner of the battle. Ten simulations of the battle for $R_0 = 300$, $B_0 = 100$, $a = 1.35$, and

[7]This law states that the sample mean or average will converge probabilistically to the mean for large sample sizes. See virtually any elementary statistics or probability textbook.

$b = 0.15$ were run with $\Delta t = 0.01$ and $\sigma = 0.7$. The graph in Figure 3.2 plots the force ratio, $F_t = R_t/B_t$, versus time for each simulated battle.

In this example, when σ is 0.0, there will be no stochastic component to attrition, and the force ratio will equal 3.0 throughout the battle until it terminates as a draw. When σ is positive, the force-ratio plot fluctuates until it becomes flat when the battle terminates. If the force ratio is below or above 3.0 at that point, Blue or Red is the winner of the battle, respectively. As demonstrated in this example, when the battle is simulated stochastically, both Red and Blue can achieve significant victories. The probability of a draw is zero.

Why Stochastic Simulation?

At the heart of a deterministic simulation of a stochastic system is the substitution of the assumed mean, μ_X,of the random variable X for the random variable itself. The stochastic properties of entities in the system are thus ignored, and it is assumed that mean equivalence holds. Mean equivalence is the concept that, for a function, f, the expected value of the function of the random variable, $E(f(X))$, equals the function evaluated at the mean of the random

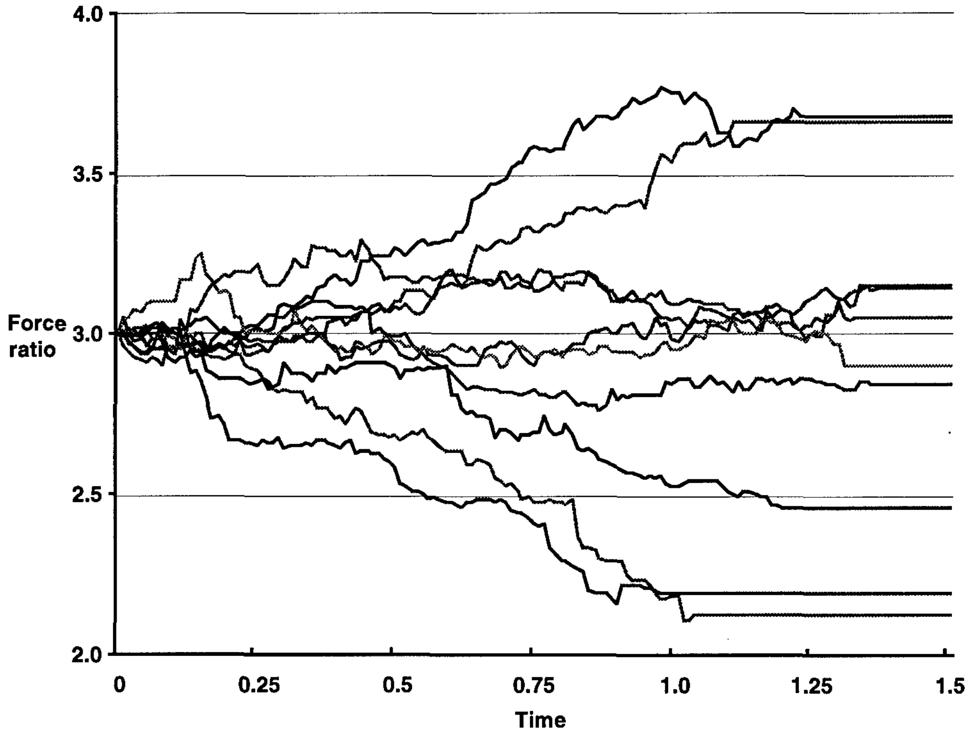


Figure 3.2—Stochastic Results of a 3:1 Ground Battle

variable, $f(\mu_x)$. But mean equivalence hardly ever applies in the simulation of a stochastic system.

The fact that the average of a repeated number of observations of a random variable may tend strongly to its mean in no way guarantees that any other function of those random variables converges to the function evaluated at the mean. The repeatability of a mean value equivalent simulation gives false security that the effects of the variance associated with the stochastic model have been eliminated.

Stated mathematically, assume that a function $f(\bullet)$ has a second order Taylor's expansion. That is, it has first and second derivatives $f'(\bullet)$ and $f''(\bullet)$. Then, if the random variable X has mean μ_X and finite variance σ_X^2,

$$E(f(X)) = f(\mu_X) + \frac{\sigma_X^2}{2} E\left[\left(\frac{X - \mu_X}{\sigma_X} \right)^2 f''(\mu_X + \xi(X - \mu_X)) \middle| 0 < \xi < 1 \right].$$

Thus, a mean equivalence assumption ignores the contributions of variance to the functions and processes of the simulation being studied unless the functions and processes are linear near the expected values of the random variables in the simulation.

As an example, suppose entities arrive at a server independently, with interarrival times , I, that are identically distributed. Further, suppose that the probability that I is greater than some nonnegative x is $\exp(-\lambda x)$. These assumptions imply that the arrivals follow a Poisson process. The quantity λ is assumed to be nonnegative and is known as the arrival rate. These entities wait in a first-in first-out queue to be serviced. The service times, T, are assumed to be randomly distributed with mean μ_T and variance σ_T^2. One measure of the congestion in the system is the number of entities waiting to be serviced. This is known as the queue length, L_Q. When the system reaches steady state, the expected value of L_Q is

$$E(L_Q) = \frac{\lambda^2[\mu_T^2 + \sigma_T^2]}{2[1 - \lambda\mu_T]}.$$

Two observations can be made about this equation. The first is well known: The queue of waiting entities will grow without bound if the expected number of arrivals during an expected service time is greater than one. The second observation is that *the variance of the service-time process contributes to the mean* of the queue length and the expected waiting time and thus to the congestion in the system. This is a common result for random variables that are the result of a sequence of stochastic processes, such as this arrival or a service-type system. The means and variances of the subprocesses contribute to the mean of the random variable of the overall process. To put this in layman's terms: variance causes congestion. We note that queues are quite common in combat models, e.g., in communication, detection, and transportation.

If this system is simulated by a deterministic simulation using expected values in place of the interarrival and service times, the length of the queue in the simulation will be infinite if $\lambda\mu_T$ is greater than one and zero otherwise. The totally deterministic simulation sees infinite congestion or none. If, instead, only the service times are deterministically simulated, the expected queue length would be an underestimate with relative error of $-\sigma_T^2/\left(\mu_T^2 + \sigma_T^2\right)$ and would grow with the variance of the service time. The partially deterministic system sees less congestion, perhaps much less congestion, than would be expected.

Another reason to avoid deterministic simulation of inherently random events is that the evaluation of $f(\mu_X)$ may not make sense in the system under study. If $f(X)$is the number of X aircraft returning from a mission that need extensive repairs, $f(\bullet)$ is not defined for nonintegers. To get around these problems, deterministic simulations generally treat entities as infinitely divisible flows rather than indivisible quanta. This then creates its own representation problems, especially if only a few, high-value entities exist in the simulation. How does 0.6 of the only aircraft carrier in a simulation function?

An alternative to simulating random processes with a Monte Carlo approach is to represent the distribution of outcomes analytically. It may be possible to predict the mean and variance of a process and use a one-time calculation of those functions rather than rerunning the many cases representing Monte Carlo trials. This is done for example in the Dyna-METRIC estimation of the aircraft availability under various maintenance and supply policies (Pyles, 1984; Hillestad, 1982; Hillestad and Carrillo, 1980). Much of the time however, it is easier to simulate the process through a series of random number draws than to estimate distribution functions or stochastic moments for complex and nonlinear processes.

The most common reason given for avoiding stochastic simulations is the number of independent runs required to achieve confidence in the results. If a campaign model requires several hours to run and scores of trials are needed for statistical confidence, it becomes a rather awkward tool for an analysis that may also need many sensitivity variations to test other assumptions. The number of runs required to achieve a given confidence level can be estimated. The estimate is computed as follows:

Suppose we are interested in forming confidence bounds for the mean value of some random variable X. We wish the length of the interval to be I. First make n_0 independent runs of the simulation with run i producing observation x_i for X. Compute the sample mean

$$\bar{x} = \sum_{i=1}^{n_0} x_i / n_0$$

and variance

$$s^2 = \sum_{i=1}^{n_0} \left(x_i - \bar{x}\right)^2 \Big/ \left(n_0 - 1\right)$$

of the observations. We will let t_k^δ denote the value that puts δ in the upper tail of the Student's t distribution with k degrees of freedom. Our confidence interval for the mean value of some random variable x will then be

$$\left[\sum_{i=1}^{n} x_i / n - \frac{I}{2}, \sum_{i=1}^{n} x_i / n + \frac{I}{2}\right]$$

after taking only $n - n_0$ more observations, where

$$n \cong \max\left(n_0, \left\lfloor \left(2st_{n_0 - 1}^{\alpha/2} \Big/ I\right)^2 \right\rfloor + 1\right).$$

If x is normally distributed, this estimate is exact. See Stein (1945). Thus, the cost of the required runs may then be balanced against the confidence desired.

If one suspects that there is variance in the system to the extent that the number of observations estimated above will be too great, the answer given by assuming mean equivalence and performing a deterministic simulation almost certainly will be subject to high uncertainty and bias. Hillestad and Owen found exactly this in their experiments (see Hillestad and Owen, 1995). Simulations of systems with high amounts of random variation tended to diverge significantly from their deterministic equivalents. In addition, the initial n_0 runs may be invaluable in exposing this possible source of variation and may help the analyst to do a better designed analysis.

The conclusion thus is clear. *If the system under study has well-behaved random variation present in the quantities of interest, there is no reason to perform a deterministic simulation. The number of runs required of a stochastic or Monte Carlo simulation in this case may be estimated and will not be great. If the system under study* *is not well behaved, this is exactly the situation in which a stochastic simulation is required to perform analysis with reasonable confidence or at least to help bound the uncertainties. The choice of choosing or building a deterministic model instead of a Monte Carlo model should depend on the nature of the system represented and not merely on expediency of processing.* TLC was implemented as a Monte Carlo simulation, but the user is provided with choices to use deterministic functions if desired.

Software Paradigms

The current choices of software structure for a campaign model are quite rich. The dynamics can be simulated using a process-interaction or event-scheduling paradigm. The entities in the simulation can be modeled as objects in various object-oriented language structures, or they can be modeled with a top-down, hierarchical structure. The architecture may strictly enforce "message passing" between objects to promote ease of multiprocessor distributed simulation or may allow more global information structures and function calls between objects. Each of these has its advantages and disadvantages as far as ease of development, modification, transparency, interfaces to data, and ability to operate as a distributed simulation. This subsection describes and compares a number of such paradigms.

Dynamic Modeling Schemes

Two schemes are commonly used to represent the dynamics of a complex system: process interaction and event scheduling.[8] A process represents the sequence of actions an entity experiences as part of the system. The event-scheduling scheme concentrates on a complete description of each event in the system. No simulation time elapses during the execution of an event. Between events, either the time-step or event-step method may be used to manage time, although the latter method is a more natural fit. With care, both schemes may coexist in the same simulation. The following discussion describes and compares these approaches.

Process Interaction Structures. Processes interact with each other by "activating," "waiting," "interrupting," "resuming," and "deactivating" themselves and each other. For example, suppose that an airplane is an entity in a simulation. To take off from a runway, it might execute the following activities in sequence: start up engine, taxi to the runway, wait for the runway to be clear,

[8]For more detail see Fishman (1978).

take off, and clear the runway for the next aircraft. Figure 3.3 presents a schematic of this process.

Time may elapse during an activity denoted by an ellipse. They themselves may be processes. Also notice that, while an airplane is taking off from the runway, it forces other airplanes to wait in a queue. At the "Clear Runway" activity, the airplane interrupts the wait of the first airplane waiting in the queue, if any, and allows it to proceed. Passive entities, those with no explicit processes, such as the runway in this example, are usually called resources. MODSIM II, used in the TLC simulation, uses the process-interaction paradigm.

Event Scheduling Structure. An event instantaneously modifies the state of the system and contains all the information and methods needed to do so. Every operation that possibly changes the system in a significant way should be represented as an event. Events "schedule" and "unschedule" other subsequent events with time possibly elapsing between them. An internal schedule handler, usually provided by the simulation language, advances from each scheduled event to the next. As an example, again consider the airplane in the previous example. The interesting events are the "Taxi Begin," "Taxi Complete," "Take Off Begin" and "Take Off Complete" events (Figures 3.4–3.7).

The "Taxi Begin" event, when executed for an airplane, will start the taxi of the airplane to the runway, compute the taxi time of the airplane to the runway, and schedule a "Taxi Complete" event at that time. No time elapses during the event processing. Figure 3.4 shows this event.

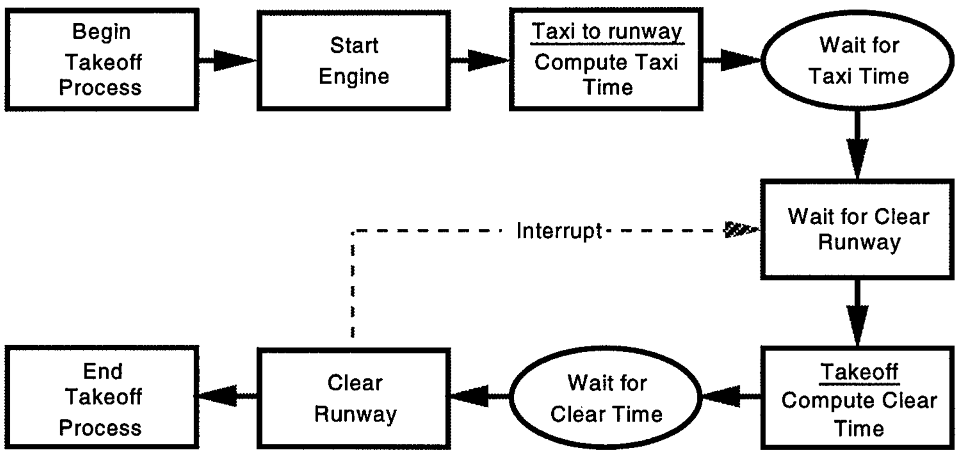


Figure 3.3—Takeoff Process

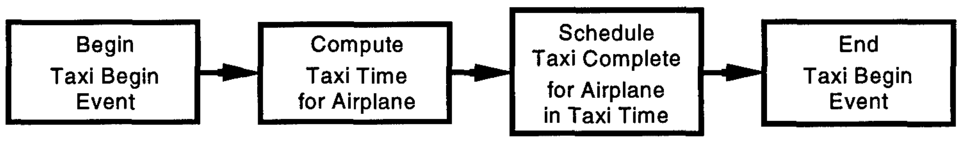

Begin Taxi Begin Event
Compute Taxi Time for Airplane
Schedule Taxi Complete for Airplane in Taxi Time
End Taxi Begin Event

Figure 3.4—Taxi Begin Event

As depicted in Figure 3.5, when the runway is not clear, the "Taxi Complete" event puts the airplane in a runway holding queue. If the runway is free, this event will allocate the runway to the airplane and schedule a "Take Off Begin" event for the airplane (Figure 3.6).

Note that no time elapses during the event processing.

The key difference between the two modeling schemes is that time elapses *during* a process but only *between* events. In process-interaction software the event scheduling and unscheduling are done implicitly by defining "waits" and "interrupts"; in event-scheduling languages, the events are scheduled and unscheduled explicitly. In our experience, the process-interaction scheme initially appears to be simpler to use in modeling and more transparent to describe. This is because the entire process is described in one routine. The event-scheduling approach does not depict the whole process involving separate event processes. Instead, one must remember its components and abstractly visualize their interaction.

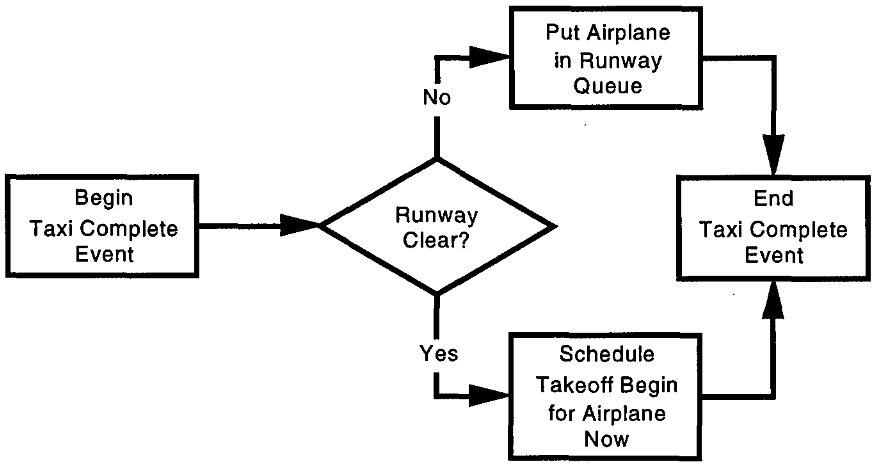

Put Airplane in Runway Queue
No
Begin Taxi Complete Event
Runway Clear?
End Taxi Complete Event
Yes
Schedule Takeoff Begin for Airplane Now

Figure 3.5—Taxi Complete Event

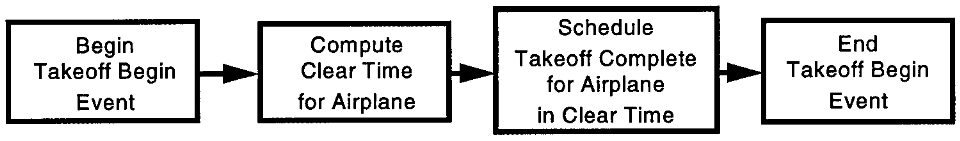

Begin Takeoff Begin Event
Compute Clear Time for Airplane
Schedule Takeoff Complete for Airplane in Clear Time
End Takeoff Begin Event

Figure 3.6—Takeoff Begin Event

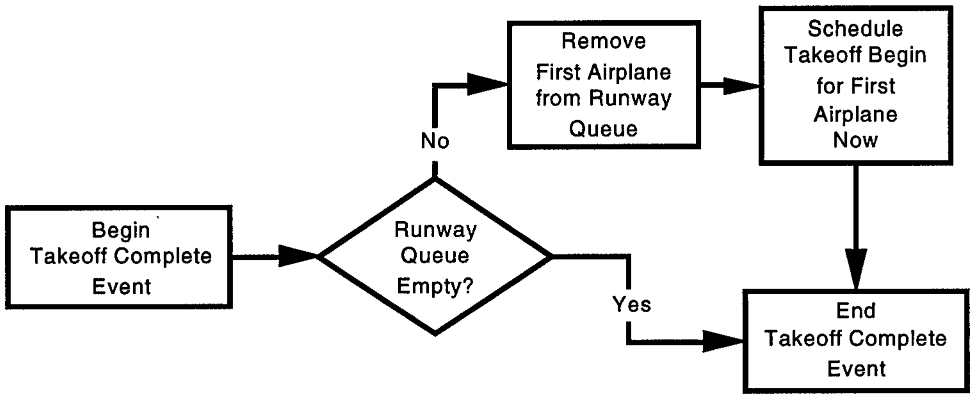

Begin Takeoff Complete Event
Runway Queue Empty?
No
Remove First Airplane from Runway Queue
Schedule Takeoff Begin for First Airplane Now
Yes
End Takeoff Complete Event

Figure 3.7—Takeoff Complete Event

But the devil is in the details. The process-interaction scheme may have less control of the flow of the simulation than the event-scheduling scheme. This is because the programmer using the process-interaction scheme frequently has little direct control over the exact timing and order of execution of the "resuming" and "interrupting" functions. Instead, he must control them using "tie breaking" routines executed by the internal scheduler of the simulation. The initial benefits of simplicity may then be lost. With care, both schemes may coexist in the same simulation. It is interesting to note that many simulation languages support both schemes and internally convert the process-interaction scheme to an event-scheduling equivalent.[9]

Object-Oriented and Hierarchical Simulation

Conventional procedural modeling concentrates on what entities do in the system, while the object-oriented concentrates on what entities are. Conventional models generally become monolithic with a master procedure overseeing the execution of subprocedures performed by the entities of the simulation. The

[9]For example, MODSIM II, a CACI simulation language, is organized around the process-interaction paradigms but implements the simulation using an event processor.

construction of conventional models is usually linear, with each subprocedure largely being a unique construct. The data for such models also take on a monolithic central role supporting the master procedure and are accessed by each subprocedure during execution. An example of a simulation language supporting the procedural modeling style is SIMSCRIPT. The use of a modular, hierarchical modeling style within a procedural model can achieve most of the benefits of object orientation. The formalism of object-oriented modeling reinforces good structured modeling habits.

Object-oriented modeling is a methodology for the design, implementation, maintenance, and evolution of a model. The system to be modeled is broken down into subsystems, sub-subsystems, and so on, which interact by passing messages among themselves. This style encourages a modular, decentralized approach to modeling. The components, the objects, are independent, localized reservoirs of knowledge. Allied with object-oriented modeling is object-oriented programming, which applies and extends the concepts of object-oriented modeling to software.

The design of an object-oriented model concentrates on two separate aspects: the internal construction of each object and the external interactions between objects. Two objects with different internal constructions are interchangeable if their external interfaces are identical. This feature allows the simulation to evolve as older objects are extended, replaced, and reused within the simulation. For example, it allows for seamless changes in resolution as higher- and lower-resolution implementations of objects are swapped. The construction of such models is inherently hierarchical. Higher-level objects are constructed from couplings of lower-level objects ultimately derived from a base of atomic objects. Simulation languages, such as MODSIM and SIMPLE++,[10] have been developed to aid in object-oriented modeling.

Object-oriented programming has four key features. The first feature is the *encapsulation* of data and code. This feature enforces the independent modular structure of the objects and data for the model. The second feature is *inheritance,* which promotes the hierarchical structure of the model. Once an object is defined, other objects may be built using that object, and these inherit its data and methods. Suppose a "Moving Entity" object is defined with a generic "Move" method and data describing the position of the object. Then both "Tank Entity" and "Aircraft Entity" objects may inherit from the "Moving Entity" and use or override the data and methods of the "Moving Entity." The third feature

[10]SIMPLE++ is a product of AESOP GmbH, Stuttgart, Germany.

is *polymorphism*. Two different objects may have methods by the same name which do much different things. The "Aircraft Entity" and "Tank Entity" objects discussed previously both have "Move" methods, and each might be asked to "Move" by another object. The asking object may not know if it is addressing a "Tank Entity" or "Aircraft Entity" but only that it is addressing a "Moving Entity." The type of movement induced would depend on the internal implementation of the "Move" method within each type of object.

These three features of an object must be used with moderation. They are not necessarily that useful in the definition of real objects. "Tank Entity," "Aircraft Entity," and "Ship Entity" objects may differ so greatly that inheriting from one common object that moves may not simplify things much. Each of these objects will inherit from a common base of simple objects, which will be augmented by data and methods unique to the entity. For example, each may have data values for latitude, longitude, and altitude but also highly differing movement methods.

The fourth feature of the object-oriented modeling style is *message passing*. Objects are treated as equals and can ask other objects for data or to perform methods using messages. Other objects have the right to refuse to release the data or to perform the requested method. This feature is quite different from the conventional procedural model, in which data are accessed from a central repository, and procedures are executed by function calls from other procedures. This is an important distinguishing characteristic between object-oriented modeling and conventional modular, hierarchical modeling. Message passing puts the emphasis on the objects rather than the procedures.

The actual implementation of message passing depends on the simulation language used and the computer environment in which the model is executed. In a distributed processing environment, actual messages may be passed between objects—sometimes over thousands of miles of network. For efficiency when using single processors, the messages may actually be function calls. On parallel concurrent processors, a combination of the two methods may be used.

The advantages, then, of object-oriented modeling are maintainability of the existing model, extendability of the model to other levels of resolution or other types of objects, reusability of existing objects, and evolvability of the model as one's understanding of the system increases. One does not begin to realize these advantages until the model is fairly complete and in use. The setup cost to achieve these advantages is a concise definition of the requirements for the model, a description of the eventual uses of the model, and a design to achieve those goals. It may actually be easier to take a procedural modeling approach for "quick and dirty," "one shot" models. And object-oriented modeling is not the only way to achieve these advantages—just a good way.

We, on the other hand, have found in our TLC research that the distributed, fractionated nature of the object-oriented style can lead to a very opaque model unless each object is well described along with its external interfaces. Each data value must be well defined and each method must be described along with any "side effects" that may be caused when executed. For example, when an "Aircraft Entity" is asked to "Move," it must be noted that several other types of objects, such as "Runway Entity" or "Flight Path," may be modified. Achieving adequate vision into such side effects is one of the most important concerns in the design and development of an object-oriented model.

Most of the legacy campaign models have been built in the hierarchical style using non–object-oriented languages. Our experience with the development of TLC in an object-oriented language indicates that the modularity adds important advantages in terms of modifiability and replacement of objects, but it is no panacea. The hard work of modeling is defining these data structures, processes, and process interactions to represent various military phenomena, and these are complex regardless of the underlying modeling and software structure.

One purported advantage of object-oriented modeling and programming is the replaceability of objects. Theoretically, as long as the replacing object provides the same public attributes and functions,[11] it should work within the simulation regardless of how those attributes and functions are actually implemented. In TLC, we created a number of alternative objects and processes, and this type of modularity works fairly well. For example, for surface-to-air adjudication, we created several alternative methods of computing attrition, from simple attrition tables to more complex time-of-exposure and probablity of kill (P_k) calculations. The real difficulty comes in creating new objects, for example, when introducing aggregate objects. All other objects in the simulation that interact with the new object are likely to need new processes, and these functions need to know the new object's attributes, etc. For example, if one introduces a company object in a simulation that previously had only tank objects and if the company is an aggregation of tanks, the aircraft need to know how to find and attack a company; command entities need to know how to issue orders to a company; intelligence objects need to know how to detect and track a company; etc. We found that object-oriented modeling programming does not make the problem of

[11]Public attributes and functions of an object are those accessible by other objects.

adding entities to a simulation any worse, but it does not really make it much easier either.

Distributed Modeling, Processing, and Analysis

Distributed simulations attempt to deal with two important problems: improving execution time and linking disparate models.

There are several approaches to improving execution time. *Concurrent* simulation involves executing components of a single simulation model on several processing units simultaneously. If the processing units reside and execute in parallel on the same computer, this type is also known as a *parallel simulation*.[12] A parallel processor may be used to do concurrent simulation. Without a parallel computer, significant time may be lost sending information between processors. The purpose of concurrent simulation is to minimize the time required to complete a single simulation run with the model.

For example, ground battles may be assigned to one processor and air battles to another. Then, battles can proceed independently and in parallel until a point in time at which the state of the aircraft affects the state of the ground battles, or the ground battle affects the air war (such as an air base being overrun by ground forces). Alternatively, the flight-generation operations at different air bases may run simultaneously on several independent processors. Other parallel schemes provide a dynamic assignment of functions to processors based on their current loading and availability.

A great deal of synchronization is required to distribute the individual computational tasks efficiently over the processors. Even so, there are usually some tasks that cause bottlenecks in the execution. Various schemes have been proposed to accomplish synchronization. Among these schemes is the "time-warp" mechanism (Jefferson and Sowizral, 1982; Marti, Kantar, and Sollfrey, 1994). This allows some parts of the simulation to advance time faster than other parts. The divergent parts get back in step only when necessary by rolling time back to the required point. The speedup achieved using this type of distributed simulation is less, sometimes much less, than the number of processing units used to make the run (Marti and Gates, 1988; Marti and Burdorf, 1990). That is, doubling the number of processing units generally reduces the time to completion by less than a half.

[12]For example, EADSIM is designed to run on four processors.

Recognizing that, for most analytic purposes, many trials are usually needed for sensitivity analysis or, in the case of Monte Carlo models, for statistical significance, the problem is not so much getting a single run to execute quickly but to get a large number of runs done. This leads to another distribution alternative known as *distributed cases*, involving execution of separate, independent runs of a simulation model simultaneously on several computers. This approach takes advantage of local area networks and the explosion in the capabilities and numbers of scientific workstations available. The individual runs must be coordinated, usually by a script (or an operator) that ensures that each computer always has a run to execute and that all the runs finally get executed. Since an entire run is performed by the same computer, no coordination within the simulation is needed. Each run usually requires some initialization that might not be needed if all the runs were performed sequentially on the same computer. The speedup achieved using distributed runs is equal to the number of computers used to make the runs, in that doubling the number of computers will halve the required time to complete the runs (except possibly for the initialization time costs). This was the approach we adopted for TLC.

Interaction Structures

Interactive or man-in-the-loop simulations allow varying degrees of human interaction with the flow of the simulation during the run, whereas a closed simulation only takes input from the human at the initiation of the run. The purpose is to provide better simulation of human decisions. Much real combat consists of the interaction of both human decision processes and physical processes. This becomes even more true at the campaign level, where command, control, and intelligence functions play such an important role. We understand relatively little about how humans actually make decisions, and we have adequately modeled even less. Thus, the appeal of interactive analytic campaign simulations is to capture innovation, risk taking, risk avoidance, and other human behavior in the decision process. Furthermore, we may well desire to capture adaptive behavior based upon incomplete or erroneous human perceptions.

There are various levels of human interaction. The first level is already employed in all good analysis efforts. The analyst examines the outputs of the model runs and will perhaps change the data for the runs to improve the credibility of the outcomes. This might consist of fixing erroneous pieces of data or adjusting the parameters on various decision processes. So, the human analyst is already in the loop.

The second level consists of stopping the model at specific points and allowing the human to enter decisions or to modify previous decisions. The model then continues using the new decisions. This approach overlooks some of the responsiveness and spontaneity of human decisionmaking, but it is often adequate for the representation of campaign planning cycles.

Finally, there is true interaction, in which the human may interrupt the machine at any time during the run to obtain information and to provide input. One way to achieve a truly interactive simulation is to modify a closed simulation so that its execution may be stopped at any time and the total current state can be saved. This is usually called "checkpointing" the model. Then, a separate interface may be used to display the saved results of the model and to allow the human to modify that state. The model can then be restarted in the modified state. Great care must be exercised to limit what the human may change and to be sure that the new state is logically consistent. It would not seem appropriate that time could be changed, for example. This presents an opportunity for existing closed models to be used in an interactive mode. The U.S. Army confederation of models used in Prairie Warrior 95 is an example of such a use (Bailey et al., 1995).

With respect to the flexibility of the underlying structure of a simulation, it is important to decide on the mode of human interaction early in the model definition. A model that permits human interaction during the course of the simulation requires that several additional features be introduced. First, some provision must be made to save the entire state of the simulation at any checkpoint or preprogrammed stop. For example, in an event-stepped model, the state and order of all events in the event queue must be saved. Second, if the human is going to be able to make decisions, the model should be capable of displaying all relevant information for those decisions, such as aggregate measures of performance at each stop. Finally, the input processes to capture human decisions must be provided along with the processes to change the necessary variables, states, and events of the model that are dependent on the human inputs. Retrofitting a closed simulation with these interactive capabilities can be quite difficult.

The use of interactive simulation has some drawbacks for analysis. Humans do not necessarily make the same decisions in a model that they do in actual combat. This can be due to the lack of finality in the outcome of the model run. No one will die due to the outcome of the model. The opposing force, at the National Training Center, fights to the death daily over the most insignificant piece of terrain. And then they fight again the next day. A second problem for analysis is that the addition of human behavior to decisionmaking in the model adds unknown variability to one of the most critical drivers for the outcomes from the model. This variability occurs between human subjects and within each subject as well. Humans have a learning curve, and their decisions for the same situation may change from one run of the model to the next. For these reasons, the introduction of humans into the model reduces the chance for the repeatability of the runs in the analysis; the design of experiments for the simulation must accommodate this fact.

The third drawback of having humans in the loop is the extra requirements for an adequate interface between the model and the human. The types of information that people use to make decisions are not necessarily the same as those calculated by the model. (The computer typically makes decisions by optimization techniques or exhaustive searches.)

Finally there is the problem of time. The decision process of human beings is based on real time; computers work much *faster*. If the decision process is sped up, the human decision may be biased toward the expedient, rather than the innovative. Alternatively, the necessity to run the simulation in real time may limit the number of analytic variations possible.

The reasons it is desirable to have interactive simulations are strongly related to the reasons their construction is difficult. TLC was implemented largely as a closed simulation with adaptive decision processes, which will be discussed in a later section.

Variable-Resolution Structures

Variable-resolution modeling (see Davis and Hillestad, 1993) refers to the construction of a single model within which analysts can readily change the level of scope or detail at which the phenomena under study are treated. Cross-resolution modeling is the linking of existing models with different resolutions. We may distinguish between various aspects of resolution. "Entity" resolution refers to the level of the objects modeled, such as battalions, divisions, and corps. "Attribute" resolution refers to the detail of the attributes of an object—for example, describing a unit by its total size and point location, as opposed to including additional characteristics that might include specific system counts and locations, weaponry, sensors, etc. Logical-dependency or "constraint" resolution refers to the level at which restrictions are enforced on the representations. That is, the actual position of a division may be required to obey the laws of Newtonian mechanics, while the battalions are only required to fit into some template around the division's location. "Process resolution" refers to the level at which various effects are calculated. An example is a sensor that uses just a cookie cutter detection range or one that incorporates other aspects of the sensor's capability (power, filtering), the target's signature, and the environment (line of sight, attenuation). Higher "spatial" and "temporal" resolution refers to the sizes of the slices used for space and time. These various types of resolution are interconnected. For example, an entity resolution that identifies individual platforms will require attribute data within a range that permits the appropriate description of these systems.

High-resolution models help an analyst to model and understand, or at least experiment upon, important underlying phenomena. The model itself captures the current level of our understanding of the phenomena and points us in directions that increase that understanding. They also help us to educate and train others. And they provide important data and insights for lower-resolution models. Low-resolution models provide us with broad screening tools that allow us to make initial cuts at the significant aspects of problems. Low-resolution models give us the "big picture" and allow us to make broad decisions in the face of uncertainty in the details of the problem at hand. They also give us a framework to judge when and if high-resolution models are applicable. Until the very nature of war itself becomes entirely predictable, we will need models at multiple levels of resolution to increase our understanding of combat (see Simon, 1981).

The process of changing resolution should be *consistent*. Suppose we have two functions, one, g, of the lower-resolution states of the model and another, g', of the higher-resolution states. These functions may be the ones that determine for their respective resolutions the next event or the outcome of an engagement or the orders to be issued at some time in the future. We define the aggregation function A such that $A(S)$ is the lower-resolution state to which the higher-resolution state S aggregates. The functions g and g' are said to be consistent if, for each higher-resolution state S, $g(A(S))$ equals $A(g'(S))$. The model is said to be consistent across levels of resolution for the functions g and g' if g and g' are consistent.[13] Although this method is simple to describe, achieving consistency in this manner is extremely difficult. Figure 3.8 diagrams the resolution consistency definition.

It is frequently desirable to disaggregate the state of a low-resolution model and use that state as the starting point for a simulation with a detailed model. For example, it may be desired to run a detailed air battle simulation starting with the numbers and types of aircraft assigned and vectored to the battle by the low-resolution model. Unfortunately, the aggregation, by definition, has lost

[13]For more details about the mathematics of aggregation and disaggregation see Hillestad and Juncosa (1993).

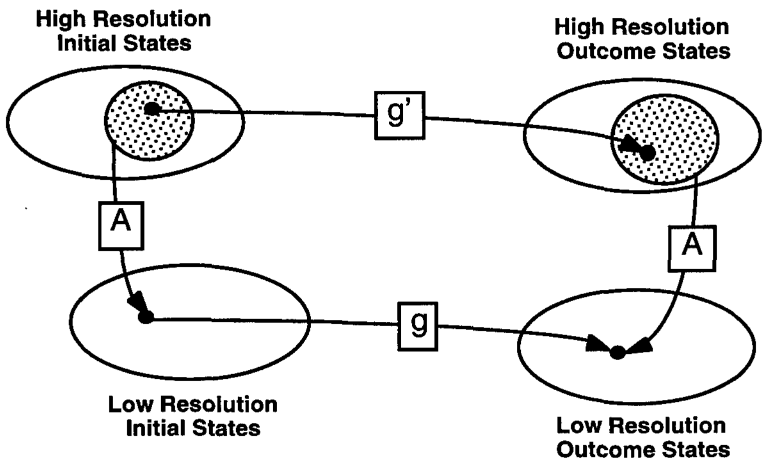


Figure 3.8—Resolution Consistency

information. There are many choices, consistent with the low-resolution state, that will lead to many different air battle outcomes. Initial positions, details of timing, command and control information available to the pilots, etc., may not be defined in the low-resolution model and must be invented to provide the initial conditions for the detailed simulation. This is one of the dilemmas with cross-resolution modeling—one must bring in information from another source if a unique disaggregation is to be made from low resolution to high resolution.

The choice of the proper aggregation and disaggregation functions to achieve consistency requires detailed knowledge with respect to the construction of the high-resolution model and about the uses to which the low-resolution model will be put.

An example of a consistent change of resolution is the high-resolution flying of virtual unmanned aerial vehicles (VUAVs) in Prairie Warrior 95 when coupled with the low-resolution model CBS. CBS units must be disaggregated to allow VUAVs to detect individual vehicles optically. The first vehicles so detected are usually on the edge of the CBS unit and usually far from the center of mass of CBS units. For artillery strikes against the CBS unit to be adjudicated properly, the firing artillery battery must be given the center of mass of the CBS unit. This is done by manual intervention by the operators of the model.

In this example, g is the artillery adjudication in CBS and A maps the vehicles seen by the VUAV to the center of mass for the CBS unit. The result of artillery strikes against vehicles at the edges of CBS units, as seen by the VUAVs, is adjudicated as if those vehicles were at the center of mass for the unit.

Another approach to variable resolution is to pick a few high-resolution states, S_i $i=1...n$, that are representative of all the high-resolution states. Then define g so that $g(A(S_i)) = A(g'(S_i))$ $i=1...n$. For low-resolution states that are not aggregations of a representative high-resolution state, the function g is defined by the interpolation or extrapolation of some appropriate functional form. In this way, the low-resolution function has been calibrated to the high-resolution function and is consistent for, but only for, the representative high-resolution states. The ATCAL[14] process used in CEM is one such process. ATCAL calibrates the attrition process of a low-resolution model, CEM, to representative killer-victim scoreboards, the high-resolution states, which are taken from a high-resolution model, COSAGE. The key to using this approach is to choose and maintain an appropriate and manageable set of representative high-resolution states to achieve the level of consistency desired.

A third approach is to start from scratch and develop a collection of integrated self-consistent functions at various resolutions. For example, the Lanchester Equations model for attrition forms one such collection. At the highest level of resolution, one may use a set of heterogeneous differential equations that linearly relates the instantaneous rate of attrition of each component of force strength for each side to the magnitude of each of the force strength components. At the lowest level of resolution, the differential equations describe the same relationship between the total force strength on each side. Hillestad and Juncosa (1993) have described conditions under which the collection of representations may be consistent across various levels of resolution. The general conditions are basic and as expected. The relationship of each component to the whole must be known, and the aggregation functions must not violate or confound those relationships. When the whole is more than the sum of its parts, extra effort will be required to understand fully the relationship of each component to the others. Variable resolution combat modeling remains an art and a science.

To summarize what is known about variable resolution, we offer the following, based on papers by Hillestad, Davis, Owen, Juncosa, and Blumenthal (Hillestad, Owen, and Blumenthal, 1995; Hillestad and Juncosa, 1993; and Davis, 1992).

1. Perfectly *consistent* aggregation across levels of resolution is possible, but the conditions of consistency may be quite restrictive, particularly for nonlinear systems.
2. Consistent disaggregation is rarely possible because information is lost in an aggregation. Only when the missing information is applied exogenously can

[14]Analysis Support Directorate (1983, 1991).

models be properly disaggregated. For example, unit positions lost in aggregation might be reintroduced using standard formation templates or by human intervention.

3. It is difficult to predict the details that will cause a difference between high- and low-resolution models. Experimentation is usually necessary. However, having done enough experimentation, it is usually possible to construct reasonably consistent, low-resolution models that capture the high-resolution phenomena.

4. Certain simulation structures may promote variable and selectable resolution more than others. Generalized network structures, event processing, object-oriented modeling, and stochastic (Monte Carlo) simulation structures appear to permit models to cross levels of resolution easier than other structures we have studied and discussed.

In TLC, we provided certain flexible structures that would permit changing the resolution of the model. The generalized network permits different resolutions of the battlefield, for example. Of course, this variation in resolution must be obtained by providing a new set of network inputs. The object-oriented programming approach also used in TLC provides the capability to create objects of difficult resolution and alternative processes for existing objects. However, these must be programmed, and the user can only select among the finite number of choices for resolution in this case. Note that, if ten objects each have two alternative choices of resolution, 20 billion interactions must potentially be preprogrammed. Clearly, achieving variable resolution is not a trivial matter.

4. Cross-Resolution Modeling in TLC

The Importance of Cross-Resolution Modeling

A key aspect of campaign models is that many of the system and force effectiveness data come from other higher-resolution models. Values for surface-to-air, air-to-surface, air-to-air, and ground force–ground force effectiveness must usually be generated by more detailed models run for a selected set of input cases. This linkage to higher-resolution models or *cross-resolution modeling* is essential to conducting high-quality analyses at the theater and operational levels for a number of reasons, including the following:

- Historical data are typically not of the right situations nor well documented or are of such an aggregate nature that they cannot be used by other than the most aggregate campaign model.[1]
- Training exercises, as a source of data, suffer some of the same problems as the historical data, with the additional issue that aspects of the training exercise may be unrealistic (sometimes on purpose).
- Weapon system test results are usually provided with too much resolution for a campaign model and may cover only a small subset of the cases encountered in a theater campaign.
- Some phenomena (e.g., the effects of terrain on acquisition) must be modeled at higher levels of resolution to be represented judiciously in theater models.
- It is difficult to compare the effects of weapon systems on campaign outcomes unless higher-resolution data about those systems are available.
- Detailed models are needed to evaluate new systems whose technological characteristics can be defined but for which test data are not available (e.g., a system with improved range but no data on expected kills).

This dependency means, first of all, that the simulation of an appropriate set of high-resolution cases must precede good analysis with the campaign model. It also requires that the linkages between these models of different resolution be valid and consistent. Unfortunately, there are serious difficulties in achieving

[1]Trevor N. Dupuy (1987) developed the Qualified Judgment Model (QJM), using historical combat data. This model attempts to predict the winner in a battle and is relatively low resolution.

such cross-resolution coupling. For years, various organizations have cited their hierarchies or families of models as the means of achieving depth and breadth of analysis of defense force structure and doctrine options. In some cases, this meant nothing more than the fact that models of different levels of resolution existed in the organization. In other cases, there were explicit attempts to feed the outputs of one model as inputs to another. Recently, with advanced distributed simulation (ADS), some of the models of different resolution have been directly linked as a single simulation.

In all cases, moving from a high-resolution model to a lower-resolution representation requires an aggregation process in which the large number of variables in the detailed model are condensed into a fewer number in the less-detailed model. Many different approaches to this have been used. For example, air-to-air combat in the TAC THUNDER (CACI, n.d.) theater model uses a set of "tuned parameters" from a detailed air combat model called TAC BRAWLER (Decision-Science Applications, 1988). They same parameters cannot be used in the TACWAR (U.S. Army, 1994a) theater model (used by the Joint Staff), which models air combat in a more aggregated manner. They also do not fit the air-to-air model used by our own TLC model, which requires "killer-victim scoreboards" from the TAC BRAWLER model. Thus, different campaign models use the data from detailed models in different forms, aggregations, and quantities.

The Difficulty of Consistent Cross-Resolution Modeling

In general, most current approaches to aggregation, while appearing to be logical, have no good scientific or mathematical basis and cannot be expected to provide consistency across the different levels of aggregation, except within very loose bounds. In papers one of the authors has published, we have defined what is meant by consistency and stated some of the requirements of consistent aggregation and disaggregation within combat models (Hillestad and Owen, 1995; Davis and Hillestad, 1993). For example, aggregations of weapon systems can be done consistently for linear combat-attrition equations only when all specific vulnerabilities and individual effectiveness values are accounted for. It is not possible to achieve consistent representation, for example, by providing relative scores of weapon systems on one side without accounting for the vulnerabilities and weapon effectiveness of those on the other side.

Even with linear attrition equations, it is generally not possible to disaggregate results uniquely from a low-resolution model into the specific effects in a higher-resolution model. The conditions under which the requirements for consistency are satisfied would be even more rarely satisfied in the nonlinear functional representations of combat in most campaign models. Thus, it must be recognized that all aggregations and disaggregations of data in combat models are approximate and that the degree of approximation error is difficult to ascertain. This is an important reason that the quantitative results from campaign models should not be taken as definitive and that there should be more research in this area.

It would seem possible to test an aggregation by taking the results from the lower-resolution model and comparing them with the same case run in the higher-resolution model. This is not as straightforward as it seems. The first problem is to derive a comparable case for the detailed model. Consider an air-to-air situation. In the campaign model, the air battle is probably represented by the number of aircraft engaging, the missions the aircraft are on, the weapons they carry, and perhaps information as to whether it is a vectored or autonomous intercept. In the detailed model, the parameters of an air battle would include the initial geometries, ranges, exact points of detection, training of the aircrews, relative altitudes, rules of engagement, preferred tactics, etc. Since many of these are not known, they must be created as part of the context for the more-detailed model. Each parameter could take on a range of values and would remain consistent with the lower-resolution model. Thus the single air battle of the low-resolution model maps into a potentially broad range of the input space of the high-resolution model. Next, consider the results from the detailed air-to-air model. For the range of inputs consistent with the low-resolution model, there is a range of outputs of the detailed model. Furthermore, if the detailed model provides stochastic outputs, this range is expanded by the possible distribution associated with each realization of inputs. The detailed model results for the plausible input range may therefore map into an even broader range of outputs. As long as the low-resolution model result falls somewhere in this range, it cannot be declared to be inconsistent with the detailed model. If the low resolution model is also stochastic, the comparison problem is further compounded because the distribution of the low-resolution model must be compared with the distribution of the detailed model. Figure 4.1 illustrates this mapping and comparison problem.

In Hillestad, Owen, and Blumenthal (1995), we have shown, by experiments with models of combat at different resolution, some of the factors that cause results to differ between models of different resolution. Spatial aggregations can create different battle geometries. Small factors, such as perception delay or nearly simultaneous firings, can cause significant differences between models when

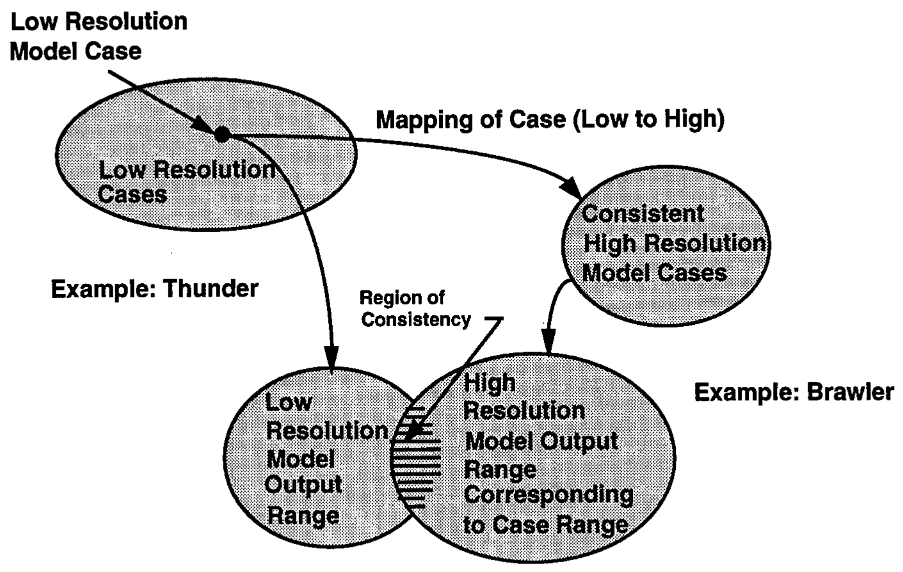

Low Resolution Model Case
Mapping of Case (Low to High)
Low Resolution Cases
Consistent High Resolution Model Cases
Example: Thunder
Region of Consistency
Low Resolution Model Output Range
High Resolution Model Output Range Corresponding to Case Range
Example: Brawler

Figure 4.1—Mappings Between the Results of Low- and High-Resolution Models

these factors are not factored into lower-resolution representations. Important phase changes of the battle may not be simulated. We also showed that, once the important factors are understood, it is usually possible to adjust the available parameters of the lower-resolution model to provide an approximately consistent representation. We have seen only one other published attempt to compare combat models of different resolutions for the same situation. The Center for Army Analysis (CAA) published a comparison of the extrapolations in the Attrition Calibration (ATCAL) close-combat computation of the CEM model with the COSAGE model outputs for the same situation (U.S. Army Concepts Analysis Agency, 1991).

One other aspect of the connection between models of different resolution is that these models usually use different time scales. A detailed air-to-air model may represent only a few minutes of battle. A JANUS-level model of combat may only simulate a few hours of close combat. A corps or theater model may represent days or even months of battle. Thus, extrapolating the data from the detailed model simplistically can create a pace of battle in the lower-resolution model that is much too high. Attrition rates from a tactical model must be drastically reduced when applied to a division, corps, or campaign model. And many other parameters do not easily scale without considering other factors. For example, it is possible to approximate the rate of movement of a few vehicles in a high-resolution model by their average speed across terrain. On the other hand, the movement of an army division is constrained by command and control factors, and the average speed across terrain yields much too high a rate. One of the important unresolved aspects of variable-resolution modeling is the determination of these scale factors and what they depend on.

Cross-Resolution Modeling in TLC

An important component of the TLC research involved how to do cross-resolution modeling, particularly for air-to-air, surface-to-air, air-to-surface, and ground-to-ground attrition. In the following subsections, we will describe the approaches used and conclude with some general observations about the process of cross-coupling models of different resolutions.

The first step in the cross-coupling process is to identify the high resolution models that will be used to provide the data for the attrition processes. In TLC, we used the JANUS model for ground combat, the TAC BRAWLER model for air-to-air combat, the Joint Munitions Effectiveness Model (JMEM) process for air-to-ground, and the RAND-developed RAND Jamming and Radar Simulation (RJARS) model for surface-to-air data. Each of these models is a higher-resolution representation of the basic attrition processes in TLC.

The general cross-coupling process involves, first, generating a finite number of high-resolution cases over a range of situations that might be encountered in the low-resolution model. This range of situations might include different types of terrain, different attack postures or missions, different mixes of weapon systems and munitions, and different assumptions about command and control. If the high-resolution model is stochastic, either a sample set for each situation might be provided or summary statistics given.

The next step in cross coupling is to match a situation in the low-resolution campaign model to one of the finite set of inputs. This generally involves some approximations and aggregations, because not all situations may be available in the finite input set. For example, some aircraft types and weapons may be approximated by data from similar types; a range of terrain types may be approximated by a single type; or several types of artillery may be aggregated into a single type.

Usually, there is also a scaling process, because the numbers of belligerents on each side in the campaign model will not always match the finite set of input situations. The results for a five-against-two air-to-air battle may need to be derived from the input results of a four-against-two battle from the detailed model. This scaling can take many forms, using relative force ratios or much more-complicated and situation-dependent relationships.

After scaling, the low-resolution campaign model uses the data from the detailed simulation in a process to estimate attrition. This low-resolution process might use the scaled data more or less directly, as with "killer-victim scoreboards," or it may use factors derived from the detailed data in a new attrition process. Figure 4.2 illustrates these different steps and the aggregation processes involved.

There appear to be two basic approaches to computing attrition in a campaign model: scoring methods and heterogeneous methods. The scoring methods tend to be the most simplified. Essentially, the various weapon systems or aircraft in a battle are aggregated into a single weapon "score" for each side. The attrition of the score is then computed from Lanchester or other equations, which depend on only a few parameters. The loss in score is attributed to each of the weapons in the battle to approximate the losses of individual systems or aircraft. The allocation of the losses is typically done in proportion to the relative number of systems in the battle. Thus, if a force is composed of 25-percent tanks, 50-percent armored personnel carriers (APCS), 20-percent artillery, and 5-percent mortars, the losses would also be distributed proportionally—25 percent to tanks, 50 percent to APCs, 20 percent to artillery, and 5 percent to mortars.[2] The parameters of the attrition formula are arrived at by various means, including the

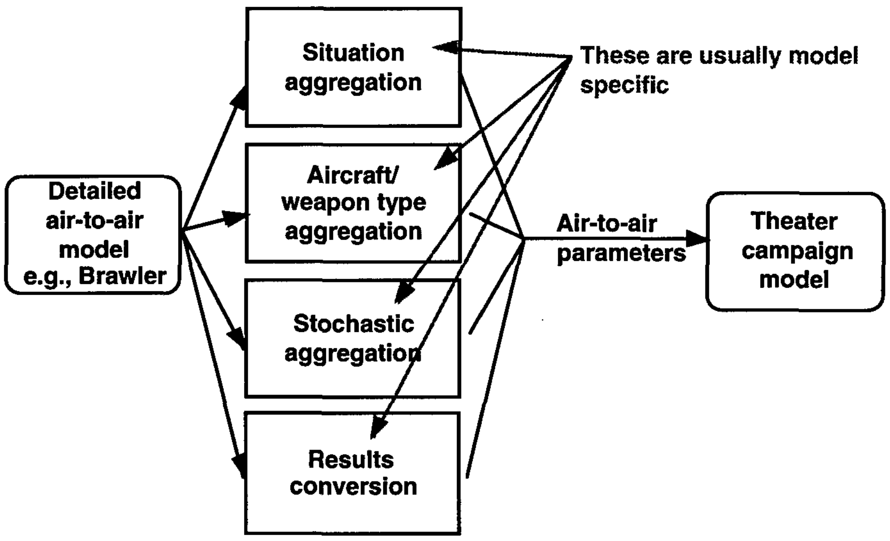

Situation aggregation
Aircraft/ weapon type aggregation
Stochastic aggregation
Results conversion
These are usually model specific
Detailed air-to-air model e.g., Brawler
Air-to-air parameters
Theater campaign model

Figure 4.2—Aggregation Processes in Model Cross Coupling

[2]The distribution of losses can take into account the situation of the battle as well. Surface-to-air defenses, helicopters, and artillery may not receive the same proportion of the close combat attrition that tanks and APCS receive. Such *situation dependent* allocation is used in the RAND JICM model.

examination of historical battles. The TACWAR model uses this approach and develops the attrition-formula parameters through the mathematical determination of the eigenvalues of the heterogeneous Lanchester equations (Anderson, Hampton, and Turley, 1980; Anderson, 1981). This approach accounts for the relative effectiveness and vulnerabilities of both sides, once one accepts the Lanchester representation. The advantage of the scoring approach is that it is less dependent on detailed model inputs, and the data are often developed from judgmental processes. The disadvantage is that, in investigating weapon systems trade-offs, the outcomes are dictated by the judgment inputs and are not based on other quantitative analysis.

Heterogeneous attrition processes do not rely on scores to aggregate weapons and forces into a single number for the calculation of attrition.[3] Although scores are often necessary inputs to command-and-control processes, they have generally been difficult to relate to more-detailed models of attrition. The basic problem with scored attrition is that the relationships between weapons of different types are lost in the aggregation. Furthermore, these relationships change over time, and only under certain special conditions can the Lanchester relationships even be maintained. Simply put, once the information about individual weapon categories has been lost through aggregation, it cannot be brought back without knowing how fires were allocated and differential attrition took place. And if the latter information were maintained, it would not be necessary to aggregate in the first place.[4]

In TLC, we attempted to avoid aggregating weapons by using heterogeneous attrition computations. In general, these start with killer-victim firing scoreboards generated from more-detailed simulations of various situations and environments. From these, both fire allocation and probability-of-kill data are extracted. Then, by adjusting these factors for situations or weapons not represented in the source information, the attrition is calculated by first allocating fire between weapon types, then determining the resulting losses. The following subsections sketch the various ground and air attrition processes used in TLC.

Ground Warfare

For its ground attrition computations, TLC uses a process we named the Calibrated Differential Equation Methodology (CADEM), a heterogeneous

[3]But a single set of scores may be used for *classes* of like systems, for example, heavy self-propelled artillery or the latest generation of tank.

[4]A methodology that attempts to remedy some of this—yet uses scores—is called Situational Force Scoring. See Allen (1992). That methodology is used in the RSAS.

ground attrition approach that is a RAND-developed extension of the ATCAL methodology developed by the CAA.[5]

CADEM bases its attrition computations on killer-victim scoreboards from various situations, with data provided by higher-resolution models and exercise data, adjusted by expert judgment to account for gaps in data coverage or to estimate attrition for hypothetical systems.[6] CADEM can also represent interactive effects between weapon systems that are dependent, as well as fratricide, suppression, shortages of munitions. CADEM allocates fire of each side against weapons of the other side on the basis of their contribution to the combat effectiveness of the opposing side in a given engagement.[7] This allocation of fire is an essential component of heterogeneous attrition computations. Because the weapon systems are not aggregated into a single score, there must be a process that determines which systems shoot at opposing systems and how much. The information in the input killer-victim scoreboards provides a starting point, but because the number and mix of weapons are likely to differ in the low-resolution model from the systems in the discrete inputs, some process of reallocation must be defined. CADEM attempts to do this on the basis of relative effectiveness and vulnerability of the various weapons.

As illustrated in Figure 4.3, killer-victim scoreboards are generated by higher-resolution models or from historical or exercise data. The killer-victim scoreboards provide high-resolution attrition information typically related to some description of the situation, the duration of battle, the current and initial levels of resources, and which systems kill which other systems. The CADEM parameter sets are calibrated to the killer-victim scoreboards and may be extended using logic, experience, or expert judgment to account for additional situational factors. The next step is the selection and adjustment of one of these extended calibration parameter sets to the particular combination of circumstances for the situation, resources, and time of the campaign model. These adjusted calibration parameters are then used to produce an attrition matrix, which is also a function of the number of resources, the situation, and time.

[5]ATCAL links division- and theater-level models at CAA and has been in use for CEM since 1983. CADEM is a closed version of ATCAL that uses differential equations to estimate outputs to the Lanchester equations that are used for computing attrition. See Louer (1991).

[6]The expert judgment is used to adjust several basic parameters of effectiveness and fire allocation for a new situation. For example, the introduction of a weapon system with a greater range would have an adjustment of its allocation against enemy systems.

[7]The CADEM methodology extends ATCAL to deal with "nonkillers" (e.g., logistics, artillery spotters), as well as "killers," i.e., units that are directly involved in combat operations. The nonkillers are targeted to the extent that they contribute to the greater overall combat effectiveness of killers against the opposing side's forces. This is accomplished by providing dependency relationships between the availability of the nonkiller and the effectiveness of the killer.

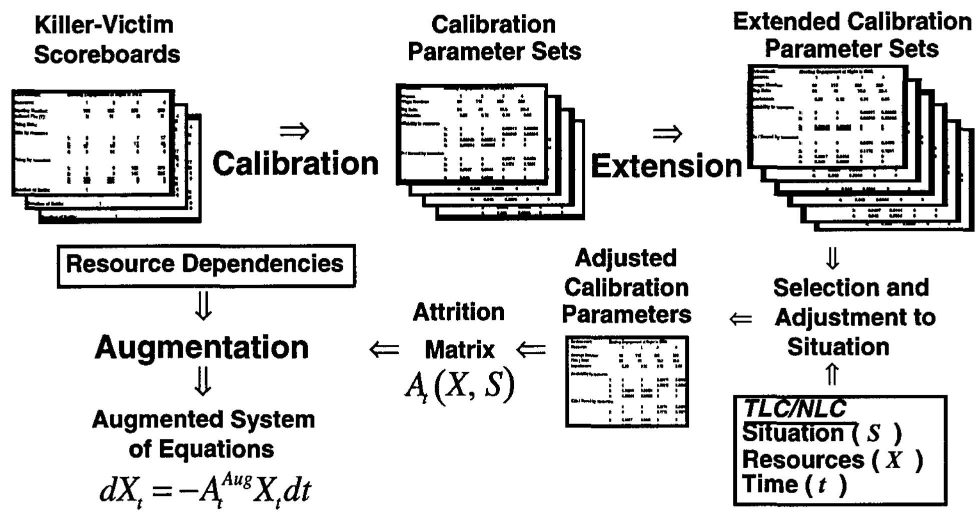

Killer-Victim Scoreboards
Calibration
Calibration Parameter Sets
Extension
Extended Calibration Parameter Sets
Resource Dependencies
Augmentation
Augmented System of Equations
$dX_t = -A_t^{Aug} X_t dt$
Attrition Matrix
$A_t(X, S)$
Adjusted Calibration Parameters
Selection and Adjustment to Situation
TLC/NLC
Situation (S)
Resources (X)
Time (t)

Figure 4.3—The CADEM Ground Battle Attrition Process of TLC

This attrition matrix may then be augmented to account for dependencies on resources (e.g., ammunition consumption, multiple systems per vehicle, or exogenous attrition) and is used in a system of differential equations that is solved to predict losses of resources as a function of time in the TLC model. The attrition matrix can be seen to be the coefficient of the derivative part from the system of differential equations. CADEM can also be used in a stochastic form, where scoreboards generated by higher-resolution models are randomly chosen from the high-resolution samples for a particular situation using Monte Carlo techniques. Appendix A describes the CADEM steps in more detail.

Figures 4.4 and 4.5 illustrate the two-sided nature of the CADEM computations and how fire allocations adapt as the number and capability of opposing systems change. CADEM allocates attrition much like the original killer-victim scoreboards but adjusts attrition to account for different mixes of weapons and capabilities; this results in reallocation of fires on both sides, with each side adapting to the other.

In Figure 4.4, there are two weapon systems on the Blue (shaded) side and two on the Red (solid) side. Weapon systems 1 and 2 are Blue systems; 3 and 4 are Red systems. It can be seen in the figure that, in the base case, weapon system 3 accounts for a large proportion of the kills of target weapon systems 1 and 2; in the second ("Halved") case, however, the rate of fire of weapon system 3 is reduced by one-half. The consequence of this is twofold. First, weapon systems 1 and 2 suffer fewer kills from system 3. Second, because weapon system 3 is less lethal to systems 1 and 2, the target priority of weapon system 3 as a target of 1

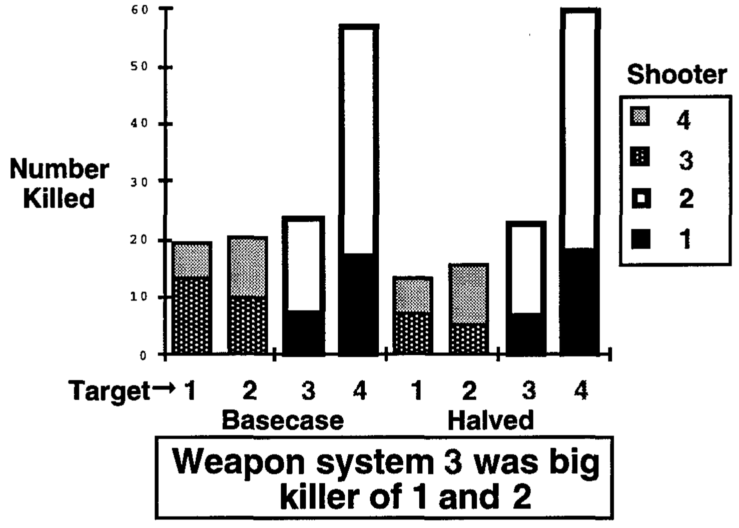


Figure 4.4—CADEM Reallocation of Fire When Capability Changes

and 2 diminishes, and fires are reallocated away from weapon system 3 to weapon system 4.

Figure 4.5 shows the differences on a relative, rather than absolute, basis. Again, the figure illustrates a high level of attrition to weapon systems 1 and 2 by weapon system 3, with slightly more of weapon system 4 killed by Blue. The interesting point in this figure is that one might have expected that more of

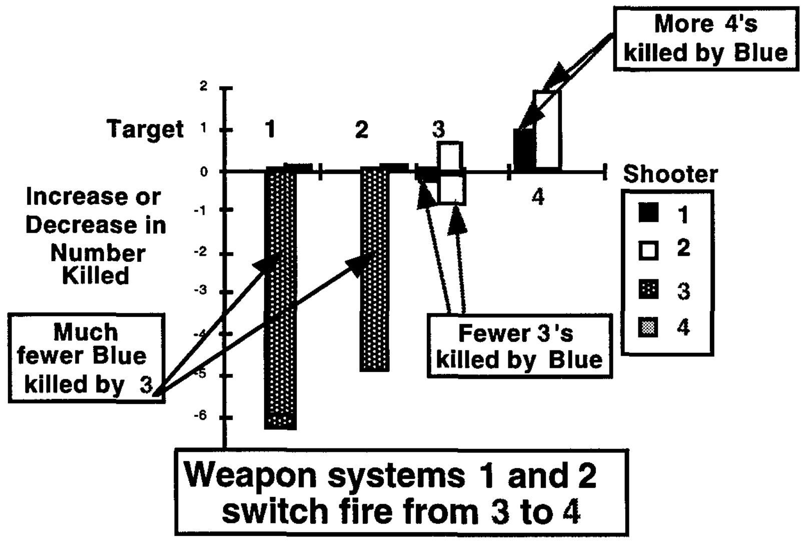


Figure 4.5—Differential Effects of CADEM Reallocation of Fire

system 3 would be killed, but since Blue reduced fires from weapon system 3, weapon system 4 became a more lethal—and consequently higher priority—target for Blue. In short, CADEM treats fire rate as an adjustable parameter, and each side reallocates its fires on the basis of the fire rate and lethality of the opposing side's weapon systems.

Advantages and Disadvantages of the CADEM Process

The CADEM approach offers a number of advantages over available aggregate attrition methodologies. Foremost is that it links to high-resolution models and maintains a heterogeneous representation of resources during battle. Its formulation and solution techniques are extensible, that is, capable of being modified. Its applicability may be extended to situations for which data are not available or are inadequate. And the system of differential equations may be augmented to capture relationships not adequately captured in killer-victim scoreboards.

This approach also has some drawbacks. It requires data for specific situations, and frequently these situations are difficult to reproduce with more-detailed models. Developing a sufficient collection may be an expensive and time-consuming chore. The ability to augment and extend scoreboards helps offset the shortage of databases, but it also allows an uninformed analyst to inject unreasonable representations into the process. In addition, CADEM may take longer to adjudicate combat than do other simpler methodologies. If proper care is taken, this disadvantage may be minimized. In comparison to the CAA ATCAL process (see Louer, 1991), CADEM requires less than 50 percent more time to adjudicate the same battle over the same time interval, although CADEM will maintain much better accuracy in some situations.

Air Warfare

In TLC, air-to-air, surface-to-air, and air-to-ground attrition computations are done by engagement.[8] Different methods are used to compute attrition for these types of combat, but, as noted above, all share an attention to situation-dependent factors and reliance on lookup tables based on detailed models and curve-fitting. In general, the air-attrition algorithms are heterogeneous in that they explicitly allocate the different systems of each side to fight explicit systems of the other side. The calculations are made situation-dependent by taking into

[8]Air attrition in TLC/nonlinear combat is based upon results from TAC BRAWLER, a higher-resolution model for one-on-one, one-on-many, and many-on-many air engagements.

account exogenous factors, such as the current state of defenses, command and control situation, and penetration altitudes. No attempt is made to score systems for aggregate attrition estimation, although some inputs defining the relative effectiveness of aircraft-weapon-crew combinations in various missions are used for assigning priorities and quantities of systems in battles. TLC air-warfare attrition methods rely on the previously mentioned detailed models for data.

Surface-to-Air Warfare

Surface-to-air warfare modeling in TLC caused us considerable difficulty for several reasons. First, we desired a variable resolution of surface-to-air combat. For many analyses at the campaign level, it would be sufficient to have surface-to-air warfare be represented merely by some situation- and region-dependent attrition factors. At the other end of the campaign resolution spectrum, we were interested in modeling the effects of dynamic SAM distribution, flight routing, defense suppression, and air-defense command and control. A second difficulty had to do with the choices and nature of the high-resolution models that could be used to develop data for surface-to-air combat. Some of these models, such as EADSIM (Teledyne-Brown Engineering, 1994), flew aircraft in missions and against integrated air-defense systems; some provided the results of individual engagements; and some provided the results of multiple raids. Each could be used, but the nature of the campaign-model surface-to-air attrition process was dependent on the selection. The third difficulty had to do with the need to develop new surface-to-air methodologies appropriate for the generalized network of TLC. We will describe the approach to these below.

Flights in TLC move along a path in the air network from the air base to rendezvous points and then through the various game board regions until they reach the target. Standoff munitions may be launched prior to reaching the target, so that the aircraft do not penetrate defenses all the way. Once reaching the terminal point (standoff distance or target), the aircraft return either by the same path or an alternative return route. The flight paths are either predefined in the input or determined by a routing algorithm.

The regions that a flight passes through while on the designated flight path can be surface-to-air regions (including SAM lines or belts), air-to-air regions, detection regions, and command and control regions. Entry into and exit from each of these possibly overlapping regions is marked by a node on the flight path network created through preprocessing the network. Figure 4.6 illustrates the flight-path and air network overlaid on the various regions.

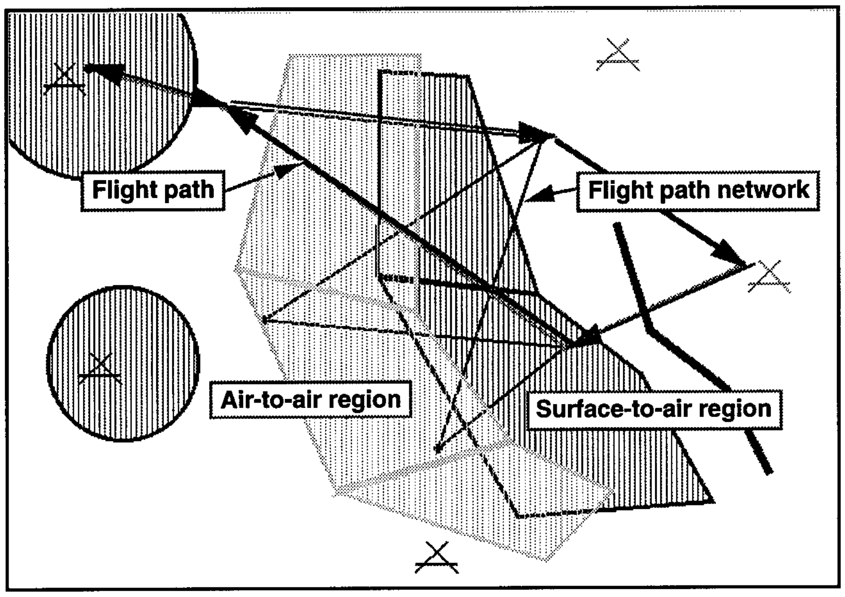


Figure 4.6—The TLC Flight Path and Regions

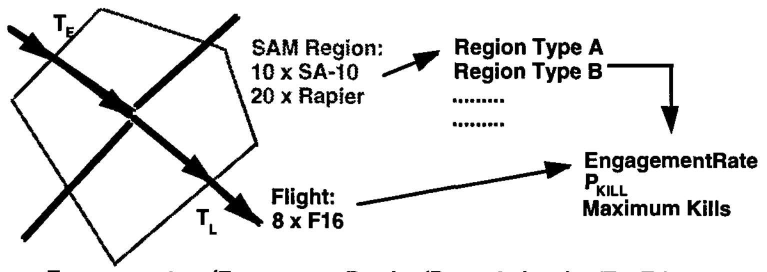
Flights of aircraft are grouped into penetration groups, with surface-to-air adjudication occurring only when a flight leaves an air-defense region, reaches its target (or standoff range), or is engaged in air-to-air combat. That is, surface-to-air adjudication is performed only when it is necessary from the standpoint of interactions with other objects. If the penetration group has previously entered a surface-to-air region, has not exited that region, and encounters an air-to-air adjudication point, the SAM attrition so far is first calculated, then lost or aborted aircraft are removed from the group before air-to-air engagements are simulated. Similarly, if the aircraft have entered and not exited a SAM region at the time the target is encountered, the SAM attrition is exacted prior to computing effects of the attack on the target.

The SAM engagement process therefore starts with a triggering event, which is usually when the flight enters an air-to-air adjudication point, launches a munition from standoff, leaves a SAM region, or reaches the target. The first step is to match the region in TLC with one of the finite set of SAM region types for which data have been input. To perform this match, TLC compares the density and mix of SAMs in the encountered region with each of the input types and selects the input region that has the closest match. In this way, as the density of SAMs in the TLC region changes because of movements of SAMs or lethal suppression, different effectiveness parameters are picked up in the inputs. The input data contain information about engagement rates and effectiveness against the aircraft types in the penetration group.

A number of other factors are used to adjust the computed attrition, including the status of the air-defense command and control (netted or autonomous), the degree to which the SAM defenses have been suppressed, the density of SAMs in the encountered region relative to the input density, the direction of the flight relative to the SAMs in the region (each region is given an orientation and arc of interest), the number of aircraft in the group (which affects detectability and also limits SAM engagements because of saturation), altitude of the penetration group, and time since entering the SAM region (time exposed). After the probability of kill of each aircraft by type in the penetration group is estimated based on these factors and the input data, a Monte Carlo draw is made for each of the aircraft in Bernoulli trials to determine the actual kill due to SAMs during that segment of the flight. This may be repeated multiple times during the penetration and return flights as more segments of the same SAM regions or new SAM regions are encountered. Figure 4.7 illustrates this process.

Variable resolution is accommodated by using more or less detail in describing the SAM regions and input data. Only a few large SAM regions might be defined for a low-resolution model in an analysis that is less interested in modeling the dynamics of the SAM battle, but many smaller regions can be used to represent the SAM and air defense layout in some detail.

Finally, we note that the important inputs from the detailed model are the engagement rate and the probability of kill, rather than attrition, even in the low-resolution representation. The reason is that exposure time is an important element of the attrition process, and because a flight may have several or many events at which SAM attrition is computed for the same SAM region, a computation based on the number of engagements leads to a mathematically



Engagements = (EngagementRate) x (Degradations) x $(T_L - T_E)$

P(Survive) = $(1 - P_{KILL})^{(Engagements)}$

Draw random number to determine fate of each aircraft

Total losses cannot exceed Engagements or Maximum Kills

Figure 4.7—The TLC Surface-to-Air Engagement Process

correct additivity, while the use of attrition does not. Put another way, if the attrition were computed simply with an attrition rate, the attrition would be different when it was broken into two parts for the same region instead of being computed once for the whole region. Use of the engagement rate avoids this anomaly.

Air-to-Air Warfare

Figure 4.8 illustrates the important objects considered in the air-to-air process. The air-to-air region is used to represent various air-control sectors or CAP regions. Associated with one or more air-to-air-regions is an air controller, who controls the allocation of fighter launches to the air-to-air region. The air controller has various alert (quick reaction or longer) fighter flights at one or more air bases. Associated with each air-to-air region are a series of detection bands—or gradients—which are used to determine whether or not a flight group traveling toward the air-to-air region has been detected. As a flight group crosses a detection line (which could be right over the launching air base in the case of long-range detection), the probability that the group is detected is computed based on its cross section and altitude and the capability of the sensor. A random draw using this probability is made to determine whether or not it was detected. If so, the air controller is notified and takes appropriate action. That action can be to launch interceptors toward an intercept point, allocate combat air patrol (CAP) aircraft to an intercept point, or do the latter and launch alert aircraft to replace the CAP expected to engage. When the flight group reaches an intercept point, it is either engaged or not. It is engaged if it was detected in time and if aircraft either on CAP or launched for an intercept would

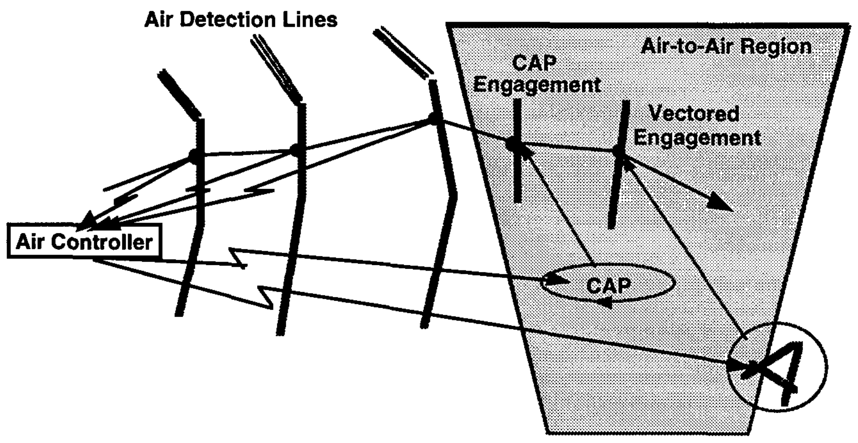


Figure 4.8—The TLC Air-to-Air Objects

reach the engagement point in time. If the detected flight cannot be engaged in time by either, the air controller projects the flight direction and attempts to engage the flight at a later adjudication point.

The air-to-air adjudication process is assessed in terms of one or more "rounds" of air-to-air engagements. Within each "round," there are three basic steps. The first step is to determine the number of interceptors that can intercept the opposing side's penetrators. The second step determines the number of escorts, sweep aircraft, or bombers that engage these interceptors. The third step determines the results of the engagements. In allowing for more than one round of engagements, explicit assumptions must be made about the operating range of interceptors surviving earlier rounds and the remaining stock of ordnance available for successive engagements. Either fuel state and ordnance may be explicitly monitored or assumptions can be made about fuel and ordnance states as a function of the preceding engagements. In TLC, we used inputs by aircraft type to define the ability of aircraft to engage in multiple battles.

Given the number of escorts, attackers, and defenders available to engage, the model first divides the battle into a set of battles that come as close as possible to matching sizes of battles represented in the input data generated with the detailed model, in this case TAC BRAWLER. In determining the mixes in these battles, the allocation uses as input relative effectiveness values in air-to-air for each combination of aircraft, crew, and air-to-air weapon. Relatively ineffective penetrators may draw only a few defenders when the defenders have a high relative capability in air defense, while advanced penetrators will be faced by many more defenders if they are available. These relative values are used in determining how many aircraft to launch in defense as well.

Once the battle composition is defined, the adjudication of losses is done by scaling the TAC BRAWLER data to match the exact composition of the force. This is done through the use of force-ratio scaling of the BRAWLER data. A force ratio higher than that for which the BRAWLER data were generated will cause a higher loss against the weaker opponent than if the force ratio were lower. The exact form of this scaling function was determined empirically using TAC BRAWLER results. After computing the probability of loss, a Monte Carlo draw for each aircraft in the battle in a set of Bernoulli trials determines the actual losses. This process is depicted in Figure 4.9.

It is important to note that the determination of which aircraft engage which other aircraft is critical in a heterogeneous attrition process, such as that described here. This process is performed hundreds and thousands of times in a

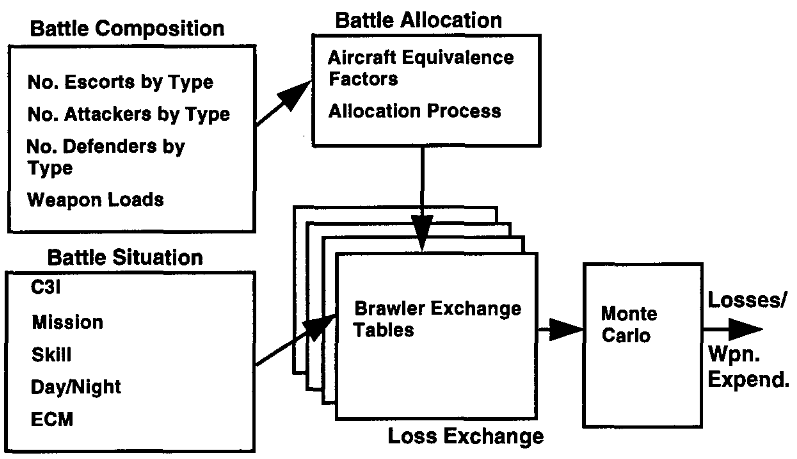


Figure 4.9—Air-to-Air Adjudication Process

large campaign and can have much more influence on the overall outcome than the specific attrition values attained from the detailed model.

Air-to-Ground Warfare

TLC's air-to-ground attrition processes (Figure 4.10) are used to estimate the effects of attack assets, such as attack aircraft (including helicopters) and cruise and ballistic missiles, on a variety of targets, including ground combat units, air bases, C^2 units and facilities, and logistics units and infrastructure. Like the air-to-air adjudication process, air-to-ground engagements are assessed in a number of "rounds," but the rounds are based upon whether the penetrators are attacking a primary or secondary target. Penetrators are assumed to go after their primary target, but if they are unable to reach it (e.g., because of weather or heavy defenses), they will instead strike a secondary target. Thus, first an assessment is performed for penetrators against their primary targets, then a second assessment is performed against secondary targets. For each, there are five steps in assessing air-to-ground engagements. The first step is to determine the number of penetrators that arrive at the target. The second step calculates the results of terminal air-defense engagements. In the third step, penetrator acquisition of the target is evaluated. The fourth step assesses damage against the target in the model. The fifth and final step determines the battle damage assessment (BDA) collected by the penetrator flight group.

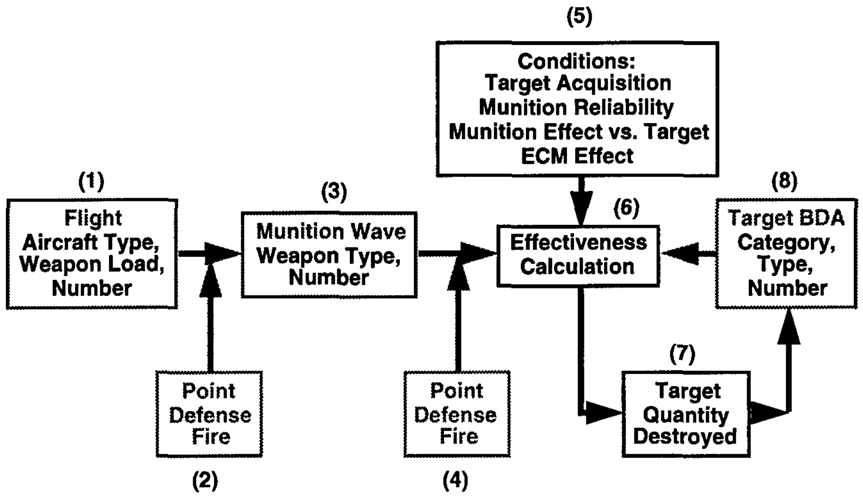

(5)
Conditions: Target Acquisition Munition Reliability Munition Effect vs. Target ECM Effect
(1)
Flight Aircraft Type, Weapon Load, Number
(3)
Munition Wave Weapon Type, Number
(6)
Effectiveness Calculation
(8)
Target BDA Category, Type, Number
Point Defense Fire
(2)
Point Defense Fire
(4)
(7)
Target Quantity Destroyed

Figure 4.10—The TLC Air-to-Ground Attrition Process

The damage assessment is based on the number of munitions that arrive on target and uses inputs from the DoD's Joint Munitions Effectiveness Manual (JMEM) to determine damage and kills of targets. Munitions are actually flown to the target as a set of flights themselves. This allows the simulation of standoff munitions and possible attrition of those weapons on the way to a target. In these data, the target definition by class defines the necessary number of hits to destroy or damage the target by each munition type and the target recovery time if the targets have the capability of recovery.

In general, there will be multiple effects of attacks on ground targets. On air bases, the shelters, runways, aircraft in the open, maintenance facilities, and crew facilities may be attacked. Strategic targets may have target damage, collateral damage, personnel casualties, etc. These possible effects are described in the input database, and damage depends on the number of aircraft arriving, weapon load types, current status of the target, and allocation of attack emphasis given in the mission-tasking order.

With respect to variable resolution, targets are defined as classes of targets, and a single location (node in the network) might represent a set of targets with multiple aim points. Input parameters describe the number of weapons needed for each aim point, and a large raid can destroy multiple targets in the group. Point defenses can be declared to defend the entire group of targets or to defend only one target aim point per point defense.

Observations About Cross-Resolution Modeling Based on the TLC Research

This section has described how TLC uses killer-victim scoreboards to establish explicit linkages to higher-resolution models and has described the heterogeneous processes associated with ground, surface-to-air, air-to-air, and air-to-ground attrition. Before going on to describe other aspects of the TLC research prototype, it is useful to reflect on the lessons and shortcomings observed in attempting to develop and use these attrition processes.

The Strict Dependency on Other Models Increases the Cost and Difficulty of Analysis with the Theater-Level Model. In contrast to models that use simple parameters, such as those involved in scoring and aggregating weapon systems, and to models that attempt to compute attrition from basic P_K factors, the strong dependency on detailed models in the processes we have described requires considerably more time and resources to develop basic data sets and to perform analysis. Many credible cases must be run with a detailed model to obtain the raw data. These must be guided by the needs of the campaign analysis and must be updated often during the course of that analysis as new factors are considered. It is not a one-time thing that produces a set of tables that can be used without changes for long periods of time thereafter but, rather, an iterative process that is analysis dependent. For example, when a new capability is to be examined, or a new situation evaluated, these must first be developed in the detailed model. Furthermore, the campaign analyst must be fairly deeply involved in defining the situations for which the detailed model is to be run and the allowable aggregations. This cost in resources and time may not be feasible for organizations operating alone with limited resources and needs for rapid turn-around in analysis. Indeed, although our goal was the ability to perform independent analysis "from the ground up," in which we would develop all our own detailed data by running a complete suite of models, we quickly discovered that it was necessary for practical reasons to rely on other organizations to help develop the data. Believing that other organizations face this same problem, we strongly recommend increased data sharing within the defense analysis community.

There Is a Need to Perform More Testing with the Various Approaches to Model Cross Connection. Although we performed some tests on these approaches to see how the campaign model results compared with the detailed model results for the same or similar situations, that testing was by no means complete and was dependent on the particular domain for which the data were generated. Broader testing of such methods appears to be rare in the defense community, but without more testing and calibration, the predictive capabilities of campaign models are even more limited than they need to be.

The Resolution of the Campaign Model Interacts Strongly with the Approach to Cross Connection of Models. We attempted to define attrition algorithms at more than one level of detail in the campaign model. For example, we created one air-to-air attrition process that used simple factors and equations to estimate the air losses crudely, and we created a second process that involved much more detail about the conditions of engagement. Each of these required a different set of aggregations of the attrition data, situation, and forces from the higher-resolution model. It should be understood that the process of cross connecting models of different resolutions is itself a modeling problem, and trying to achieve different levels of resolution in the same model often compounds the difficulty of model cross connection.

Real-Time Cross Connection of Models of Different Resolution Is Probably Not the Solution. It might be suspected that, rather than running all the cases needed for our heterogeneous approximations, it would be possible just to connect the two models and actually run the detailed model for each air battle arising in the campaign model. The problem with this, besides the possibly excessive run time, is that all the mapping problems must be foreseen and resolved so that a situation attained in the campaign model is appropriately mapped into a high-resolution case, with data not available in the campaign model and needed for the detailed model created in such a way as to be at least consistent with the campaign situation. We doubt this can be achieved in very many cases. The lack of an analyst interface between the models can also lead to undetected errors when, for example, some boundary conditions not accommodated in the higher-resolution model cause it to create arbitrary results.

Community Sharing of Data and Testing of Approaches May Increase the Practicality and Quality of Model Cross Connection. This brings us to our final point. A common effort by the defense community involving sharing of data, critiquing approaches to data generation, and testing the cross connections in hierarchies of models could reap dramatic benefits. In the current environment, there appears to be too much redundant effort at developing databases and too little testing and review of approaches. Both the quality and timeliness of analysis suffer from the current organizational barriers and lack of scientific approach to such data development and model testing.

5. Modeling Maneuver in TLC

Background

An important goal of the TLC research was to investigate the modeling of maneuver in a campaign model. Maneuver is defined in the U.S. Army's field manual as follows:

> **Maneuver** *Place the enemy in a position of disadvantage through the flexible application of combat power.* Maneuver is the movement of forces in relation to the enemy to gain positional advantage. Effective maneuver keeps the enemy off balance and protects the force. It is used to exploit successes, to preserve freedom of action, and to reduce vulnerability. It continually poses new problems for the enemy by rendering his actions ineffective, eventually leading to defeat. At all levels of war, successful application of maneuver requires agility of thought, plans, operations, and organizations. It requires designating and then shifting points of main effort and the considered application of the principles of mass and economy of force. At the operational level, maneuver is the means by which the commander determines where and when to fight by setting the terms of battle, declining battle, or acting to take advantage of tactical actions. Maneuver is dynamic warfare that rejects predictable patterns of operations. (Department of the Army, 1993, p. 2.5)

Historically, maneuvers and countermaneuvers have been critical factors in battles. A few well-known examples include Schwarzkopf's "left hook" during Operation Desert Storm, the Inchon landing in Korea, Patton's countermaneuver against the southern flank of the initially successful German surprise maneuver in the Battle of the Bulge, and the rapid and unforeseen moves of the Germans in the early stages of World War II against the French and the British. Ground commanders attempt to "outflank" the enemy, and sea commanders have historically attempted to gain an upwind advantage over their opponents. The advantages come in terms of focusing force against vulnerable components of the enemy, achieving surprise, and defeating a force piecemeal, that is, creating battles that are to the advantage of the one maneuvering in the most favorable way. Maneuver is also involved in defense when the defender moves his forces in such a way as to avoid battles advantageous to the attacker. In this way, a weaker force can defeat or defend against a stronger opponent. Maneuver involves the timely movement of forces and fires and an information war between the opponents. Denying information about oneself, deceiving the enemy, and good intelligence about the opponent are essential characteristics of maneuver. Commanders talk about "getting inside the decision cycle" of the enemy, meaning that information, decisions, and movement of one's forces occur more quickly than the other side can respond to.

During the Cold War, campaign models mostly neglected the maneuver aspects of warfare and performed dynamic balance assessments between opposing forces lined up in the pistons of the NATO defensive layer cake organized around corps boundaries. To be fair, games played with hex-based models, such as IDAHEX (Olsen, Candan, and deJijs, 1985), in which human decisions directed the play of the forces, did accommodate the maneuvers of the game players. However, most constructive models in which decisions were automated used the piston paradigm. As the Cold War drew down and conventional arms control agreements began to endorse deep cuts in the forces, the interest in maneuver warfare increased. The U.S. Army began looking at the "nonlinear" battlefield and various analysts began to look at options for defending with forces "below the operational minimum," in which contiguous defense lines could no longer be supported across NATO's Central Region. The end of the Soviet Union as a threat and of the NATO–Warsaw Pact face-off has led to a new security environment filled with considerably more uncertainty about where, when, and against whom our forces might be applied. The probable need to deploy and the possible short warning means that our forces may be considerably outmanned or outgunned in the initial stages of a contingency, and this in turn increases the desire to turn the tide with superior maneuver. The U.S. advantages in intelligence collection and command and control assets are seen as one means to achieve this superiority.

Analysis of such capabilities and scenarios means that a model capable of examining the current security environment should have the means to examine the important aspects of maneuver—the competitive "nonlinear" movement of force and fire, the role of information, and the decision cycle. This is what we attempted to examine in the TLC research.

Clearly, the ability to automate all decisions with respect to maneuvering forces in a model is beyond the reach of our current knowledge and capabilities, as is the simulation of most complicated human decision processes. The design of ground-force maneuvers can easily be more complicated than chess (which machines now do well, but the rules are highly constrained and the information is perfect). It is not very likely that we could get a model to invent good maneuvers totally from thin air. We therefore limited the research to attempting to develop a means of significantly adjusting an input operational plan based on an ongoing information battle. Of course we also had to define a model that could move forces or fires as desired in an arbitrary operational scheme. The remainder of this section describes our approach, which we simply call the "C^2 Planner."

Ground-Force Operational Plans

An operational planner typically engages in a highly iterative process in the development and evaluation of his plan. At the outset, the planner will define objectives in terms of position, time, and attrition; define paths to all objectives; allocate ground and air assets to achieve the objectives in both the close and deep battle; evaluate (as well as replan, if necessary) the resulting plan; and play both attacker and defender in a fair, unbiased manner, attempting to anticipate moves and countermoves. During an operation, the planner will then dynamically evaluate progress toward objectives; reallocate ground and air assets in response to the evolving situation; and whenever necessary, modify objectives, priorities, and paths. These are what the TLC C^2 Planner attempts to simulate.

An operational plan or concept consists of objectives in terms of locations, times, and level of attrition, along with the resources perceived to be available to each side. In addition, avenues of advance are specified, as well as the hierarchy and priority of the attacks. An operational plan has been formally described for purposes of implementation in TLC, and part of that formalization accommodates the network description of movement adopted by this model. A typical operational plan in TLC consists of the elements illustrated in Figure 5.1:

- A *path* is an ordered list of nodes and arcs denoting the general path along which ground units are to proceed.
- An *order* is a path along with various parameters that guide or control movement along the path, based upon environmental, operational, and doctrinal considerations, such as night-only movement, maximum road speeds, and specified start times.
- An *avenue* is an order with additional information pertaining to an attack, including the start time and the deadline for the attack along that avenue, and the side (i.e., attacker or defender) whose resources will be allocated and monitored. The attacker will attempt to move along the path, reaching the end by the completion time for the attack, while the defender will oppose the attacker.
- An *attack* is a collection of avenues. Attacks may be classified as *main*, in which case they are designed to achieve the principal objectives of the operation; *supporting*, in which case they are designed to support the

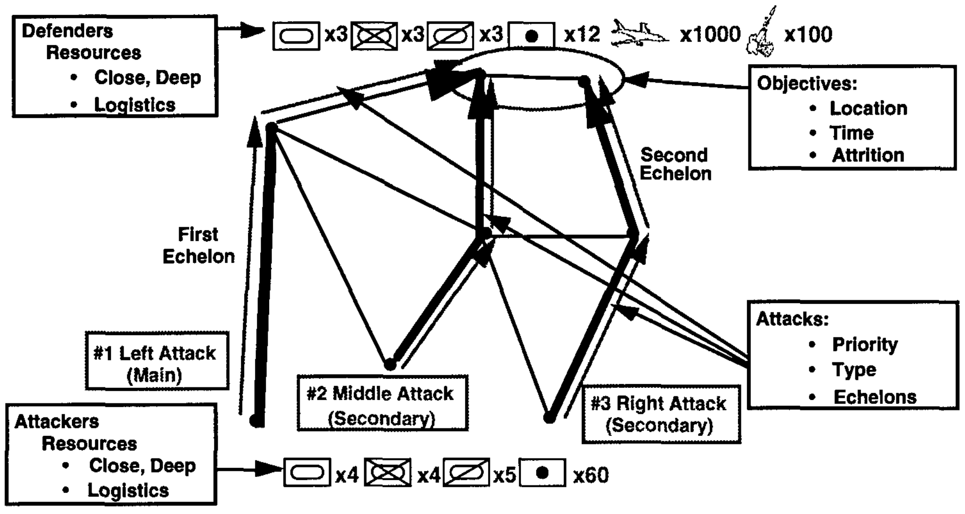


Figure 5.1—Elements of the Operational Planner

associated main attacks; or *holding*, in which case the attacks have only limited objectives designed to occupy enemy resources.

- An *attack plan* is an assignment of resources for each side to the avenues of an attack. One side's resource assignments represent a best guess at the adversary's intentions. The attack plan must specify the attacker and the defender for each avenue.
- An *attack objective* is the collection of end points for the avenues in an attack and their associated deadlines. To achieve an attack objective, the attacking side may be required, by the deadline, to occupy or control the elements of the attack objective, to attrit enemy resources by a certain amount, or to limit the attrition of his own resources.[1]
- The *situational evaluator* is the method by which the likelihood of an attacker's achieving an attack objective at any time during the attack is assessed. If an attack has adequate likelihood of success, the attacker is said to be "substantiated."[2]

[1]The attack objective may be represented by a specific point or points on the TLC network or a less deterministic region specified by an arc of the TLC network.

[2]For example, in an extension of the probability that one side will withdraw first using the Lanchester square law, we might compute the "required rate" (RR) of advance needed to achieve the attack objective, the "force ratio" (FR) of attacker resources to defender resources for the attack, and the "width ratio" (WR) of avenue frontages to the width of the attack (10 km for main, 5 km for supporting, 1 km for holding). One method for computing the likelihood of the attacker achieving the attack objective—of the "probability of substantiation" (PS)—uses the equation:

$$PS = 1 - EXP\,[-30 * (1 + (FR - 1) * WR) / (18 + RR)\,]$$

An *operational plan* is thus a collection of coordinated and prioritized attack plans.[3] The operational plan may specify a *Reserve* for use at the discretion of the operational commander, who then monitors each attack plan and reallocates resources between attacks, and the reserve using the iterative operational planning process.

Evaluation of such a plan requires estimating whether it is feasible at each level of the hierarchy of command in terms of the time, position, and attrition objectives, and if not, how to reallocate forces or change objectives to make it so. This evaluation requires estimating how an enemy would utilize its air and ground forces, calculating movement rates, and estimating attrition. This in turn requires estimation functions for these attributes of the plan.

TLC was set up to (1) provide a graphical means to input alternative operational plans easily, (2) accept various estimation functions for attrition and movement, and (3) automate the adjustment of a plan in terms of reallocating resources and changing the priority of the components of the plan. After specifying the beginning positions of units with the TLC GUI, the planner is free to specify in detail how all combat and other resources will be used. Alternatively, the model enables the user to perform such allocations with the assistance of either artificial intelligence–based scripts or optimizing algorithms. To perform its reallocation of ground and air resources, the C^2 Planner uses decision tables and network optimizers for tactical-level decisions and interacts with the air-planning algorithms to allocate interdiction assets. TLC's air allocation methodology is discussed in Section 6.

The TLC C^2 Planner

TLC's operational level C^2 Planner assists in the performance of the operational planning functions by evaluating the likelihood of success of an operational plan and allocating reserve ground forces and interdiction assets to maximize the likelihood of achieving the highest-priority objectives. It also serves planning for either attacker or defender or both. During the execution of a plan, it performs similar functions, monitoring the operational plan and adjusting priorities and the allocation of ground and air resources to improve the prospects for success of the plan and reviewing decision tables and network optimizers for tactical-level decisions. This process is illustrated in Figure 5.2.

The attack is "substantiated" if PS > 0.90. Other mathematical approaches may be found in Taylor (1983), especially pp. 270–300. See also R&D Associates (1990), pp. 5.89–5.96.

[3]An operational plan can also be a defense plan in which attack objectives are fallback points and might include a counterattack plan.

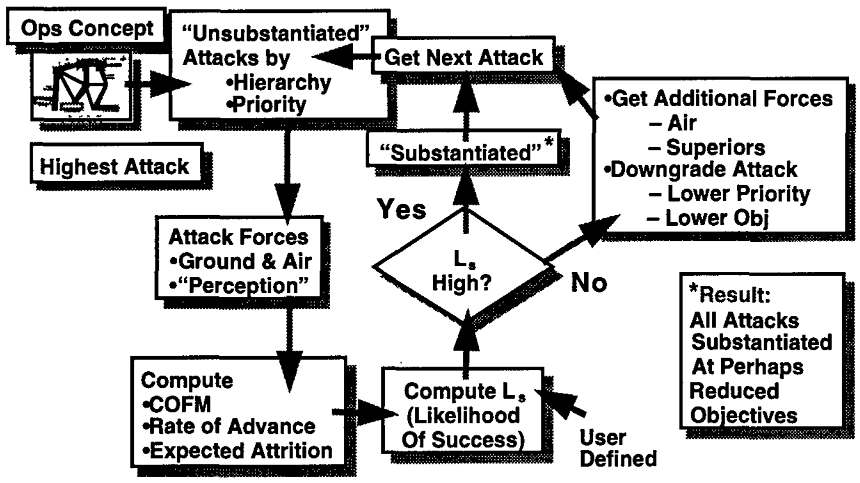

Ops Concept
"Unsubstantiated" Attacks by •Hierarchy •Priority
Get Next Attack
Highest Attack
"Substantiated"*
Yes
•Get Additional Forces – Air – Superiors •Downgrade Attack – Lower Priority – Lower Obj
Attack Forces •Ground & Air •"Perception"
L_s High?
No
*Result: All Attacks Substantiated At Perhaps Reduced Objectives
Compute •COFM •Rate of Advance •Expected Attrition
Compute L_s (Likelihood Of Success)
User Defined

Figure 5.2—The TLC C^2 Planner Validates and Adjusts an Operational Plan

Given a concept of operations, as previously described, the C^2 Planner ranks the attacks and defenses in the concept first by hierarchical level and by priority within hierarchy. All these attacks and defenses are initially termed "unsubstantiated." The C^2 Planner is based upon a network and uses algorithms derived from the COFM of the former Soviet Union to reevaluate plans and reallocate forces dynamically.[4] This approach to planning (and dynamic replanning) effectively constitutes a top-down planning algorithm in which a commander makes conservative assumptions about the allocation of enemy forces and then, based on position, time, and attrition objectives, uses a set of rules or tables to evaluate the probability of success (correlation) of the plan. When the probability is high, the plan is judged to be successful and to be "substantiated." Otherwise, more forces are applied to the plan, or the plan is made less ambitious by lowering priorities or goals. Once a plan is validated at the highest level, supporting subordinate plans have to be substantiated as well.

If the subordinate plans are not substantiated, an iterative process takes place until they are. A plan is always substantiated eventually, but the substantiation adjustments may lead to such a downgrading of objectives that the forces do nothing but hold or retreat.

In the past, RAND simulated this process for the allocation and reallocation of Warsaw Pact ground forces in the earlier combat models and has continued to refine and generalize the approach in subsequent research, including that on this model (Parker and Wegner, 1989). In TLC, the feasibility of objectives is

[4]For a brief review of Soviet thinking on COFM, see Hines (1990), especially pp. 191–193.

evaluated by examining the avenues of attack. If the evaluation suggests that intermediate phase points along the avenues of advance can be reached by their specified deadlines, the attack is executed. On the other hand, if the evaluation suggests that phase points cannot be reached by the deadline, alternative approaches to meeting the objective are assessed.

The TLC C^2 Planner may be used in two-sided optimized simulations to represent opposing commanders' adaptive allocations or reallocations of resources in response to those of the opponent. In short, for each time period, each attack is evaluated in priority order to determine whether it is substantiated—i.e., whether there is a high probability of its success. If it is, the operational planner may widen the attack by adding another avenue to the attack and/or committing resources from lower-priority attacks (or operational reserves). If an attack plan cannot be resubstantiated, the planner may reduce its priority or adjust its attack objectives. This adaptive approach brings to the operational planning process far more dynamic—and sometimes counterintuitive—allocations of combat and noncombat resources.

Tactical Movement in TLC

The movement of ground forces through a generalized network in TLC was briefly described in Section 2. The movement along axes is dictated first by the C^2 Planner, which defines which units are to move to which axis, at what time, and in what order. The ground maneuver units for the particular resolution defined for the situation at hand are treated as objects that, for purposes of movement and engagement, are broken down into units of the next lower resolution. Thus, if the maneuver unit resolution chosen is a division, the units move and engage as brigades. Figure 5.3 illustrates this breakdown of units. Each subunit has a number of resources, including weapon systems and logistics.

Each of these lower units of resolution is moved as a point object on the network. That point object occupies the capacity of a segment of the network until it has passed, at which time it releases the capacity. If the subunit takes all of the capacity of the segment, no other subunit can move until the first unit releases the capacity. Units can move in an administrative formation strung out on the route or in combat formation and off road. Each arc segment has both on- and off-road capacity (which may be nearly infinite, as in a desert, or much more constrained, as in swampy or mountainous terrain. When units move into contact range, the engagements are defined by the specific subunits within range and the formation of those subunits. Several individual battles between different

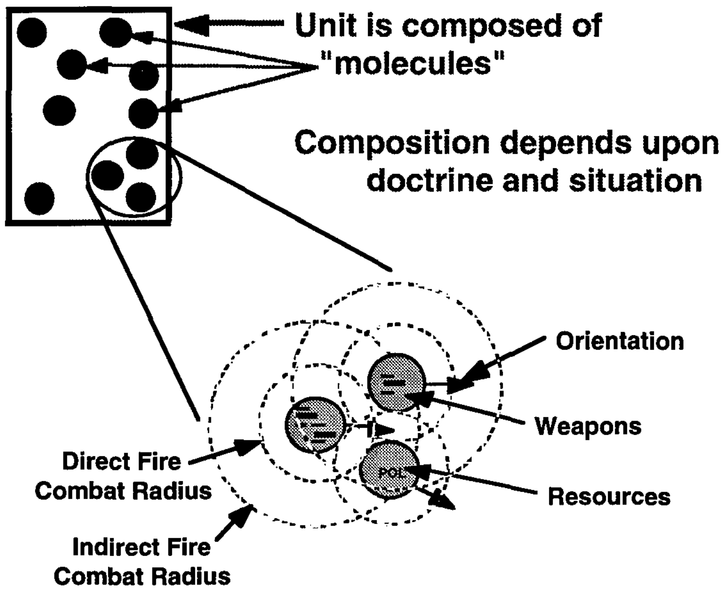

Unit is composed of "molecules"
Composition depends upon doctrine and situation
Orientation
Weapons
Resources
POL
Direct Fire Combat Radius
Indirect Fire Combat Radius

Figure 5.3—Maneuver Units and Molecular Subunits

groupings of subunits may occur as a result of the formations, confluence of arcs of the network, and timing. Figure 5.4 illustrates such engagements.

Favorable battle situations are therefore dependent on the ability of a maneuver unit to get more of the combat resources of its subunits into an engagement than an opponent; this, in turn, depends on the ability to maneuver through the network more favorably and faster than the opponent and to achieve relative surprise—basically to be more maneuverable. In the generalized network representation of movement, units can be attacked on the side, on a flank, and from multiple sides. In the battle adjudication this affects the number of weapons that can be brought to bear and the effectiveness and the composition of the weapons (of the subunits) in the battle.

Routing through a network for a unit that is directed to one location from another can be direct, representing a cross-country move in which the shortest direct path is taken (a temporary arc is added to the network in this case), or via the existing network. The latter case is the most common. The moves along the network can be scripted or defined by a special shortest-route network algorithm. This algorithm finds the shortest, "least dangerous" route to the desired location using arc distances and ownership of the arc (friendly, neutral,

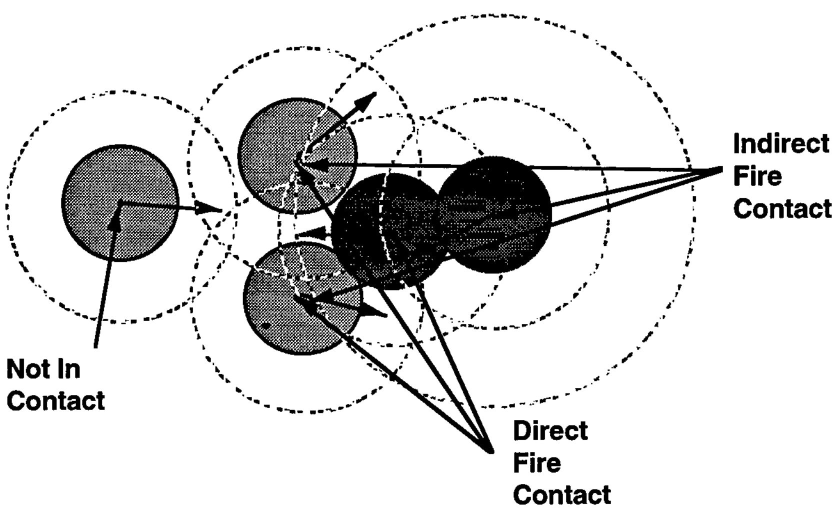


Figure 5.4—Molecular Subunits Forming an Engagement

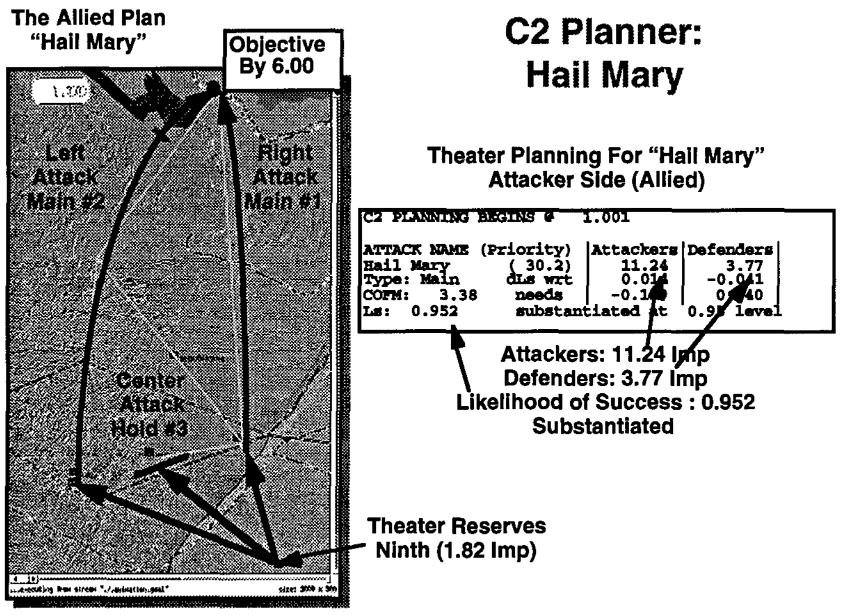
or enemy) and, in effect, placing a larger penalty on routes that must move through enemy territory than on those that move through friendly territory.

Illustration of a Simple Plan

Beginning with Figure 5.5, the next several figures illustrate the functioning of the C^2 Planner in a simple case. The concept of operations for the allied C^2 Planner involves an overall attack, "Hail Mary," with three subattacks.

The highest-priority subattack is "Right," which attacks directly toward the objective for "Hail Mary" at the top of the image. This is a main attack. The next highest priority subattack is "Left," which is also a main attack. The "Left" subattack swings out to the left and then toward the objective. The lowest-priority attack is "Center," whose job it is to hold the center of the line. The time limit to reach the objective is five days, hence, D+6. There is one ground unit in reserve, the "Ninth," and no allocatable air resources. A conservative substantiation level of 95 percent has been chosen. Thus, with a COFM of 3.38 and a likelihood of success of 95.2 percent, the overall attack "Hail Mary" is substantiated.

Figure 5.6 shows that the "Right" subattack is next in the queue for evaluation. The intelligence network has detected the "Republican Guards" enemy unit



C2 Planner:
Hail Mary

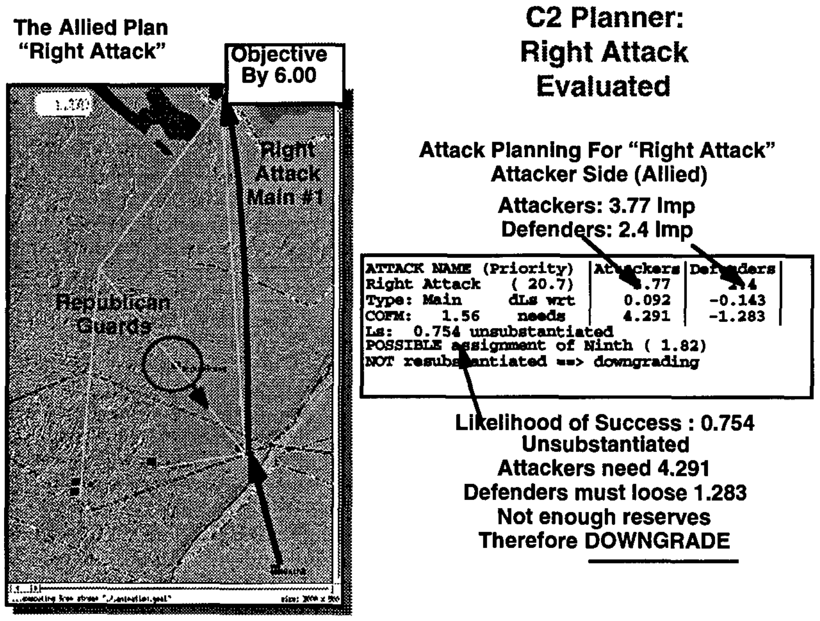
Theater Planning For "Hail Mary"
Attacker Side (Allied)

```
C2 PLANNING BEGINS @   1.001

ATTACK NAME (Priority) |Attackers|Defenders|
Hail Mary       ( 30.2)|   11.24 |    3.77 |
Type: Main     dLs wrt |   0.014 |  -0.041 |
COFM:    3.38   needs  |  -0.1[illegible] |  0.[illegible]40 |
Ls:  0.952      substantiated at  0.9[illegible] level
```

Attackers: 11.24 Imp
Defenders: 3.77 Imp
Likelihood of Success : 0.952
Substantiated

Figure 5.5—C2 Planner: Hail Mary



C2 Planner:
Right Attack
Evaluated

Attack Planning For "Right Attack"
Attacker Side (Allied)

Attackers: 3.77 Imp
Defenders: 2.4 Imp

```
ATTACK NAME (Priority) |Attackers|Defenders|
Right Attack    ( 20.7)|    3.77 |    2.4  |
Type: Main     dLs wrt |   0.092 |  -0.143 |
COFM:    1.56   needs  |   4.291 |  -1.283 |
Ls:  0.754 unsubstantiated
POSSIBLE assignment of Ninth ( 1.82)
NOT resubstantiated ==> downgrading
```

Likelihood of Success : 0.754
Unsubstantiated
Attackers need 4.291
Defenders must loose 1.283
Not enough reserves
Therefore DOWNGRADE

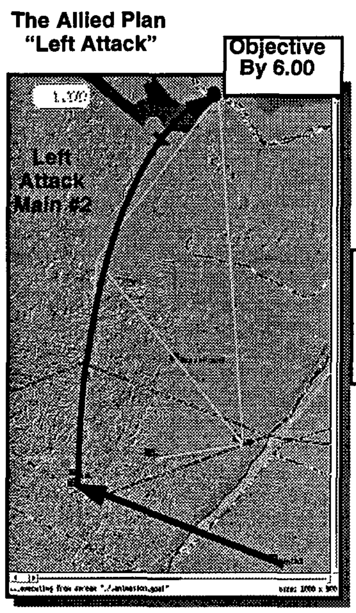
Figure 5.6—C2 Planner: Right Attack Evaluated

moving to oppose this attack. Thus, when the likelihood of success is evaluated, it is much too low. In addition, insufficient force is available to resubstantiate the subattack. Thus, the "Right" subattack is unsubstantiated and is downgraded.

In Figure 5.7, the "Left" subattack is also unsubstantiated, but it can be resubstantiated with the assignment of the "Ninth" reserves. If air resources were available to destroy 0.269 importance score (Imp), the subattack could also be resubstantiated by the allocation of air resources. The value passed to the air planner attacking the "Republican Guards" would be 18.8 percent per importance score destroyed.

In Figure 5.8, when the "Ninth" is assigned to the "Left" subattack, it becomes resubstantiated. When executed by the model, the modified attack plan achieves its objective.

If the "Republican Guards" move against the "Left" subattack instead, the C^2 Planner allocates the "Ninth" to the "Right" attack as shown in Figure 5.9. The value of timely intelligence to the planning process is demonstrated in this example. In fact, without a reactive planner, the value of such intelligence is difficult to access.



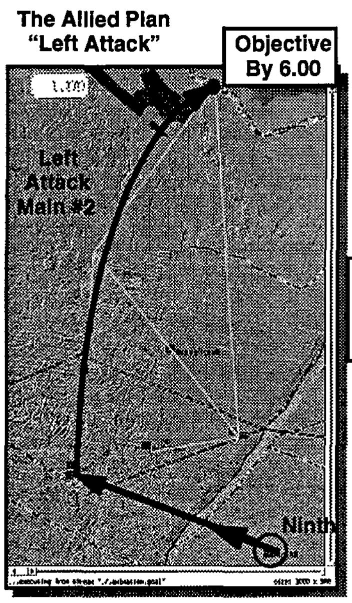
C2 Planner:
Left Attack
Evaluated

Attack Planning For "Left Attack"
Attacker Side (Allied)
Attackers: 3.77 Imp
Defenders: 1.21 Imp

```
ATTACK NAME (Priority) | Attackers | Defenders |
Left Attack    ( 20.5) |     3.77  |     1.21  |
Type: Main     dLs wrt |    0.060  |   -0.188  |
COFM:   3.13    needs  |    1.082  |   -0.269  |
Ls:  0.902 unsubstantiated
POSSIBLE assignment of Ninth ( 1.82) to
resubstantiated by Hail Mary
```

Likelihood of Success : 0.902
Unsubstantiated
Attackers need 1.08
Defenders must loose 0.269
Assignment of Ninth will resubstantiate
Note: Interdiction of 0.269 Imp of Defender would also resubstantiate with marginal value of 0.188 per Imp.

Figure 5.7—C^2 Planner: Left Attack Evaluated



**C2Planner:
Left Attack
Resubstaniated**

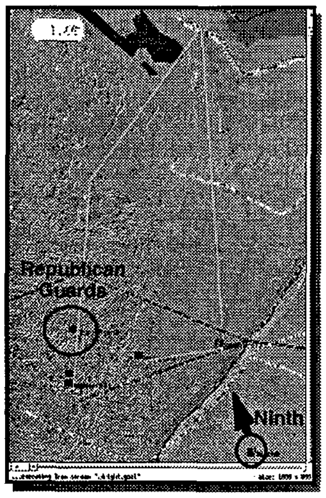
**Attack Planning For "Left Attack"
Attacker Side (Allied)**

**Attackers: 5.59 Imp
Defenders: 1.21 Imp**

```
ATTACK NAME (Priority)  |Attackers|Defenders|
Left Attack     ( 20.5) |     5.59|     1.21|
Type: Main     dLs wrt  |    0.020|   -0.091|
COFM:    4.64   needs   |   -0.741|    0.184|
Ls:  0.968      substantiated at  0.95 level
ASSIGNED Ninth ( 1.82) to Left Attack
```

**Likelihood of Success : 0.968
<u>SUBSTANTIATED</u>**

Figure 5.8—C^2 Planner: Left Attack Resubstantiated



**C2Planner:
Right Attack**

When the Republican Guards advances against the Left Attack the Ninth is allocated to the Right Attack.

Figure 5.9—C^2 Planner: Right Attack

Some Observations About Modeling Maneuver on Generalized Networks

When modeling maneuver, the tactical doctrine for each side in the conflict must be considered. The network and the methods used in the simulation to represent actions of molecules on the network control the quality of the doctrinal representation. The doctrine for troop control; the response of ground units to air attack; and the actions of units before, during, and after battle are examples of these methods. In addition, sensor management and detection processes and the procedures used to avoid detection should be incorporated. Since these methods will usually need to be examined and analyzed during an analysis, they should be input as data to the model with as little "hard wiring" as possible in the code. The use of a decision system based on compiled and interpreted decision tables seems to be indicated. Various learning and optimization techniques may be used as long as speed of execution is not adversely affected to a significant degree.

After the game board is constructed, paths through it must be determined. A path is a collection of game board nodes and arcs that describes the route a molecule will take from one location to another. Paths can be entered manually, but frequently, the fastest, shortest, or least-threatening path is desired. The computation of such paths may be needlessly time-consuming if performed during a simulation run. This is because a large number of redundant computations will be performed within and across simulation runs. Preprocessing of paths eliminates the redundant computations and aids game board construction by revealing unrecognized problems. For example, the least-threatening path may cross through a known military defense area because of a missing air defense region, as shown in Figure 5.10. After the missing air defense region is added, the least-threatening path avoids the area, as illustrated in Figure 5.11.

Again, a GUI aids in the visualization of the paths. After the game board and paths have been constructed, the operational concept must be defined for the semiautomated planners. The ultimate objective of these planners is to assign air and ground forces to attacks or defenses to maximize the likelihood of achieving prespecified objectives. At the heart of each attack and defense are one or more paths.

Separate networks should be used for each side to limit the information each side gets from the other. The mere existence of possible "left hook" paths on a Southwest Asia game board would reveal much about possible U.S. plans. In addition, the use of separate networks for each side allows the parameters of a

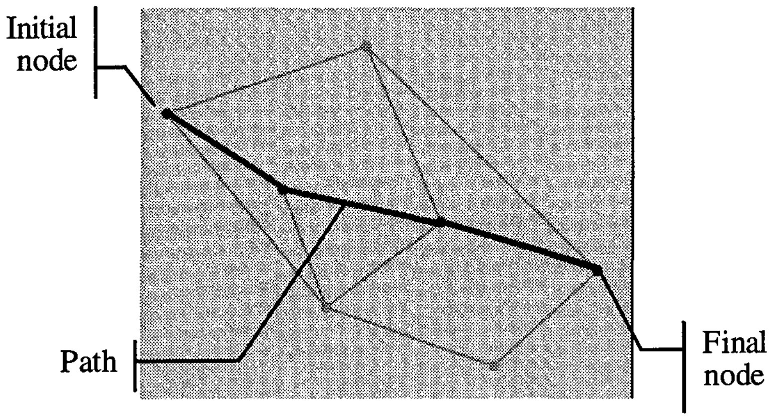


Figure 5.10—Least Threatening Path—No Defenses

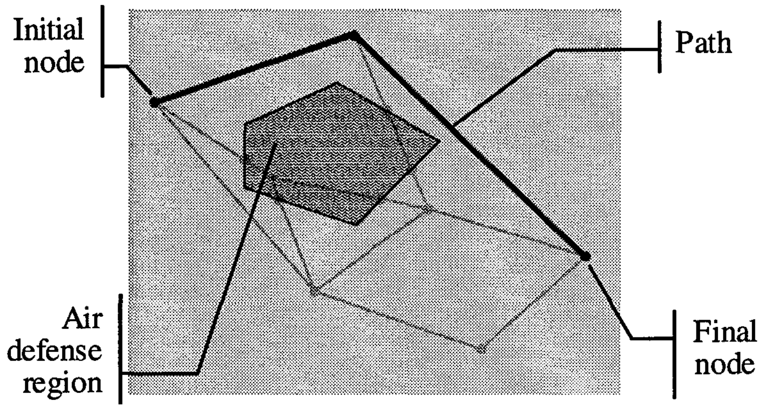


Figure 5.11—Least Threatening Path—Defenses Identified

network to be adjusted according to the doctrine and tactics for the side planning with that network. In this way, the network an operational concept uses serves to encapsulate the information pertaining to the operational concept.

To carry the point further, each operational concept should be built by separate teams of experts. These teams would build the concept by repeatedly gaming the outcome of the concept and adjusting it as necessary. A GUI aids this development process by serving as a "graphical sand table." Note that this process may require improvements to be made to the network, paths and, perhaps, the game board features. A neutral team, the referees, then reconciles the operational concepts and associated networks to ensure proper interactions of forces on the game board. The referees may point out problems to the individual teams and rectify those problems if the purpose of the analysis is not violated.

6. Automated Adaptive Resource-Allocation Processes

Background

Decisionmaking and command and control issues dominate much of campaign analysis. At this level of warfare, the problem is often not how much force there is but how and when it is used. Despite the importance of this problem, we believe it remains the least understood and most inadequately modeled part of most campaign-level simulations. Some models simulate nearly perfect battlefield intelligence, and each side has unrealistically perfect knowledge of the other side's positions, forces, and capabilities. Furthermore, many of the models implicitly model nearly instantaneous allocations and reallocations and, in effect, assume a nearly perfect command and control process. On the other hand, many models use scripted force allocations as inputs to the simulation. These scripts do not adapt to changes in situation during the course of the simulated battle. In general, most models do not adequately reflect the competitive nature of decisionmaking between opponents on a battlefield.

Many believe that the U.S. dominance of the information assets in a campaign should provide a distinct advantage over most of our potential opponents.[1] The ability to view opposing forces and maneuvers, to disrupt the enemy's planning and execution, and to feed him disinformation while denying him the same capabilities against us should provide for significant force multiplication. We would like our models to show this and to help make decisions about the acquisition of various intelligence assets. Unfortunately, models that do not react to information and the uncertainty about information by adapting strategies and resource allocations cannot show the value of those assets and capabilities.

TLC has been designed to cope with the need for a model that can support the development of adaptive strategies for policy analyses. There are other reasons such a capability is important for policy analysis. Adversaries are reactive and goal oriented. As a consequence, real strategies are highly interactive, and the outcomes of these interactions are often highly unpredictable—the consequences of actions may result in quite counterintuitive responses. An adaptive model can

[1]See, for instance, U.S. Army (1994b).

assist the analyst in recognizing critical uncertainties regarding threats and opportunities. Another reason is that policy analyses often need to consider how the "best" allocation of resources with respect to time (when, in what sequence), space (where), and function (close air support [CAS] as opposed to battlefield air interdiction [BAI]) for fighter aircraft, reconnaissance as opposed to attack for helicopters, direct as opposed to indirect fires, etc.) will affect the value of those resources. Often, adaptations that an opponent can make will reduce the effectiveness of those resources, and a new capability should often cause a change in one's own strategy.

The Decisions in a Campaign Model

Many levels of decisionmaking are represented in a campaign model. At the highest level are the expression of campaign objectives and the commander-in-chief's strategies for achieving these objectives within the constraints of the campaign. These include the phasing of the campaign and the emphasis on broad operational objectives: strategic targeting, air superiority, and the ground battle. Objectives of the campaign include the areas to defend, territory to capture or recapture, and punishment to inflict. There may also be constraints on rules of engagement, on acceptable attrition, and on operating with allies. Most of these must be declared exogenous to the model, although it may be desirable to examine the influences of several variants.

At the next tier, the operational level, are the decisions about force allocations, priorities and timing for achieving these objectives. These include how multirole aircraft are allocated to missions over time, how the ground forces are reinforced, and what types of targets to emphasize over time. The operational plan of maneuver—the creation of phase lines—for the ground forces is also at this level.

We group the decisions at the lower levels as tactical tasking. These include the allocation of weapon fires of a ground unit against the weapons and forces of the other side, the allocation of fighters against penetrating bombers and escorts, the choices about which airfields or strategic targets to attack in a unit's fragmentary mission orders, and the allocation of surface-to-air fire of a coordinated (netted) set of SAM fire units. Many of these tactical choices also have very large effects on the outcomes in the campaign model. A less-than-optimum allocation of defending aircraft (too few or too many launched) against penetrators throughout the course of a campaign may make a big difference in the overall air superiority battle. Simulated ground units firing against the wrong targets may alter the entire course of the ground battle; CAS aircraft allocated against the wrong ground units will depreciate the air support effectiveness.

Many of the effects of tactical command and control decisions are to reduce or increase force-on-force effectiveness in battles. Although most of the effectiveness numbers used in a campaign model are likely to come from higher-resolution models, tactical decisions must also be made in the campaign model that affect the ultimate performance in battles. In TAC THUNDER and TLC for example, the actual breakdown of an air battle in terms of which defenders engage which escorts and bombers depends on internal rules and algorithms that have embedded assumptions about what is known about the composition of the enemy force, missions, and the relative capabilities of the various aircraft in air-to-air battles. The compounding of many of these tactical decision rules over time leads to a strong and sometimes hidden impact on the overall campaign.

In reality, the operational and tactical decisions in a campaign depend strongly on the other aspects of the C^4I system. The intelligence process provides the assessments about forces, positions, capabilities, and intent of the other side, based on various types of sensors (including human), communication of the sensor information (not always in real time), and computer and human fusion and dissemination of the information. The command process decides and uses the communication net to send the operational and tactical decisions to the various force elements in the campaign. The control process monitors and reallocates forces and fire based on new information and the overall plan. Improvements to C^4I add to the information, improve the intelligence assessment, speed the transfer of information, improve the allocation of forces, and make the control system more adaptable. Disruption of the C^4I of a side will cause the forces to operate in autonomous, uncoordinated modes; operate with erroneous information about the enemy; or not be able to respond as fast to enemy actions. It is these types of effects that one hopes to capture in a campaign simulation of C^4I and the related decisionmaking processes.

It should be noted that we often have great difficulty modeling humans in high-level decisionmaking activities. Humans use much more than the current state of the battle to make force allocation decisions; they use their training, experience, and knowledge of specific enemy leaders to make good or bad decisions under the pressure of time and the real threat of a malevolent enemy. The following subsection compares and contrasts in more detail the various approaches to modeling human decisionmaking.

Approaches to Simulating Human Decisions

Emulation of all the human decisionmaking within a campaign model is difficult at best, and some aspects are so poorly understood that they cannot currently be represented with any degree of realism. It is only in extremely structured game situations, such as chess, that we have had any success in getting computers to compete adequately with people at a high level. Over the years, many approaches to modeling the decisionmaking in a campaign model have been tried with varying degrees of success. Some approaches attempt to use humans interactively with a model, rather than to model the decisionmaking itself. Others attempt to capture a set of "rules" developed from decisionmakers. Still others attempt to capture the objectives of the decisionmaking process and to use algorithms to adapt to situations that fixed rules do not address.

Because there can be both bad and good decisions in actual combat, there is a question of whether always to use the best rules or optimal decisions or whether to degrade the decisions somehow to be more representative of reality. This, of course, depends on the question at hand and the approach to analysis. If the analysis is to evaluate a new system, it might be useful, in an a fortiori experiment, to assume that the allocation of that system is the best that can be done. If the system does not then pay off, it probably will not when less-than-optimal allocations of the system are used. We may also want to find out how the system fares against an enemy who makes the best response to its use, that is, we make the assumption that we should not evaluate systems against a "dumb" enemy.

The approaches we will discuss in the remainder of this subsection are

- man-in-the loop simulation
- scripted decisionmaking
- rule-based/decision table methods
- tactical algorithms
- value-based methods
- learning methods
- searching algorithms
- objective-driven optimization and gaming methods.

Man-in-the-Loop Simulation

Interactive or man-in-the-loop simulations allow human beings to interact with the flow of the simulation during the run, while a closed simulation only takes input from the human at the initiation of the run. Using man-in-the-loop simulation, we may capture innovation, risk taking, and risk avoidance in the decisionmaking process. We can see the effects of adaptive behavior based upon incomplete or erroneous perceptions upon the outcome of a campaign.

The use of interactive simulation has its drawbacks. Humans do not necessarily make the same decisions as part of a model that they do in actual combat and, furthermore, are not consistent in their decisions at different times. The addition of human behavior to decisionmaking in the model adds unknown variability to the results. This variability occurs between different human decisionmakers and within each decisionmaker as well. As humans learn about the consequences of their decisions, they may change those decisions in subsequent runs of the model. Thus, the introduction of human decisionmakers reduces the repeatability of the outcomes of individual runs in an analysis. This may or may not be desirable, depending on the purpose of the analysis.

Another drawback of humans in the loop is the extra requirement to construct an adequately realistic representation of the state of the model and of the decisionmaker's options. Lack of realism in the interface may bias the decisionmaking process. For example, if the decisionmaker knows that the information he is presented is ground truth with no uncertainty, he may be willing to take risks that he would not take in actual combat. Some decisionmakers will also "game" an unrealistic model to produce favorable effects. A player who discovers that his aircraft carrier cannot be sunk may devote all his assets to offensive operations rather than allocating some to defensive operations.

There is also a problem with timing. The decisionmaking process of human beings is in real time, but computers can work through a simulation much faster. If the decision process is accelerated, the human decision may be biased toward the expedient rather than the innovative. Or bad decisions may result from the inability to keep up with the information flow. Finally, there are important cost considerations in getting the players together.

Scripted Decisionmaking

To simulate human decisions in a model with consistency and repeatability, one may script the decisions. One technique for obtaining such a script is to execute the model with humans in the loop and record the decisions made. This record, or a synthesis of several such records, can then be replayed for subsequent runs of a closed model. A drawback of this technique is the lack of adaptability. No matter what the situation, General Custer will always go to the Little Big Horn and await the pleasure of Chief Sitting Bull and Chief Crazy Horse.

An alternative technique to obtaining a script is to execute the model iteratively. At each decision point, a decision is chosen that is viewed as best by one or more experts in the light of what has been learned from previous runs of the model. By viewing the consequences of decisions in prior runs, mistakes may be avoided. Proceeding in this manner, experts make choices for each side in the simulated conflict, either cooperatively or competitively. This technique usually produces very conservative decisions. Now Custer will never advance toward the Little Big Horn.

The major problem with scripted decisionmaking is that it lacks spontaneity and adaptability. Suppose a script has been developed in circumstances in which the effectiveness of air-to-ground munitions is very poor. The suppression of sorties of air-to-ground aircraft by attacking air bases supporting those aircraft would not be rewarding and would be avoided in the script. In addition, aircraft capable of air-to-ground attack would be assigned to other missions if at all possible. Now suppose that the existence of vastly improved air- to-ground effectiveness was postulated as a form of sensitivity analysis. The script would not take advantage of the improved munitions, and the contribution of that improvement would not be properly evaluated unless the scripting was redone for that specific case.

Rule-Based and Decision-Table Methods

To introduce adaptability into scripts, one may instead adopt a set of rules. Each rule is a decision to be taken when specific conditions are met. Rules may either be developed by experts or inferred from observations of human decisionmakers in actual combat, exercises, or man-in-the-loop simulations. The rules are usually organized into decision tables. Decision tables list the conditions and for each condition describe the decision to be taken. Suppose General Custer's decision to advance or retreat from the Little Big Horn depends on two factors: the number of Indians observed and his own strength. Table 6.1 might be the decision table for General Custer's decision.

Such tables allow the desired decision for each circumstance to be provided as data rather than being coded in the model. Decision tables may cause calculations to occur or may refer to other tables or to themselves iteratively, thus allowing fairly complex decision rules. Anabel is an example of a state-of-the-art language that supports such decision tables.[2] Procedures written in Anabel may

[2]In an unpublished draft, Robert H. Anderson, H. Edward Hall, and Norman Z. Shapiro of RAND have described features of the RAND Anabel language and programming environment that support verification and validation.

Table 6.1

General Custer's Decision Table

Number of Indians Observed	Own Strength	Decision
Less than 100	At least 1 Rgmt	Advance
Less than 100	Less than 1 Rgmt	Retreat
Between 100 and 1,000	At least 2 Rgmts	Advance
Between 100 and 1,000	Less than 2 Rgmts	Retreat
More than 1000	Any	Retreat

be called by simulations written in other languages, such as C. An interesting feature of Anabel is that the decision tables may be entered as data and may be modified during a run. The latter ability allows for a degree of human interaction in the decision process.

One problem with decision tables is that one must anticipate all the conditions and all the possible decisions for all conditions. Given the large number of state variables in a campaign model, this can be a very difficult and time-consuming task. A second problem is that the objective of the decisionmaker is implicit and not always apparent in the rules. This can reduce the explanatory aspects of the model and limit the ability to make consistent changes to the decision rules.

Tactical Algorithms

Several lower-level, yet still important, decisionmaking processes may be implemented directly in code. This is usually done for the sake of increased execution speed and because the process is intimately connected to the algorithm involved. In campaign models, such processes are usually considered to be at the tactical level. For example, the tactical decisions for fire allocation in close combat are usually embedded in the close battle attrition process for a campaign-level model. In TLC, the CADEM process described previously allocates fire in close combat in one of three ways. The first is simply to allocate fire according to the density of weapon systems on the battle field. If one weapon system is twice as prevalent as another it will be fired upon twice as often. The second approach is based on a set of likelihoods provided by the user. In the third approach, weapon systems fire at other weapon systems according to an internally generated set of priorities. The priorities are determined according to the ability of each weapon system to kill other weapon systems.

Other tactical decisions in a campaign model affect the setup of air battles in terms of, for example, which aircraft fight which other aircraft, how many aircraft are launched to defend against a raid, and which ground forces to attack by which aircraft. Many of these decisions are embedded in the model code and are controlled by input parameters. Relative capabilities of aircraft in air-to-air and desired air-to-air force ratios may be inputs that affect the air-to-air tactical allocation.

The tactical decision rules embedded in a campaign model, although often hidden, have a very significant effect on the outcomes in the model. A rule that always assigns too few defensive aircraft to an incoming raid will create a large bias in the course of thousands of air battles. Similarly, the ineffective allocation of ground fire by weapon type will strongly affect the campaign outcome after many days of ground warfare. The influence of tactical decisions should be examined in the course of a campaign analysis.

Value-Based Methods

In value-based methods, the outcome state of each possible decision is estimated or predicted, perhaps with a "look-ahead" method. That state is evaluated and assigned a number according to some metric, with better outcomes usually getting larger numbers. Sometimes the outcomes are assigned several numbers: one for the value of the outcome with respect to each of several metrics. The decision problem then becomes the problem of finding the decision with the *best* evaluation. If several decisions yield outcomes with equivalent best values, a tie-breaking scheme is used, perhaps even a coin flip. Many computerized chess-playing programs use this method. In differential value-based methods, the current state is also evaluated and assigned a number based on one or more metrics. The decision taken then is the one that maximizes the improvement in outcome divided by the *cost* to execute the decision. The key to all these methods is in defining a good evaluation scheme for the states of the simulation. In addition, an efficient procedure to search the space of outcomes is required.

Learning Methods

Value-based methods also allow the simulated decisionmaking process to "learn." As decisions are made, the process stores a numeric evaluation or "fingerprint" of the state of the system when the decision was made, as well as a fingerprint for the state of the system at some time afterward, representing the effect of the decision.

When a decision results in a bad outcome, that decision is pruned from the set of decisions to be considered if the same situation were to present itself again. After many computer runs, perhaps very many, the process can search its history of fingerprints and decisions to determine good decisions to take and bad decisions to avoid, wherever it encounters situations having fingerprints that it has previously cataloged. Notice that any type of simulated decision procedure may be used to initialize and assist the learning process. When the process encounters situations it has not seen before, it makes its decision with decision rules, "best value," or "best differential" value methods. Modern chess-playing programs using such schemes have become nearly invincible. Good human players sometimes beat the programs by confusing the computer—creating circumstances the program recognizes and then making creative decisions and causing situations that the program has not encountered previously. This demonstrates the strength and weakness of learning algorithms: They do not make the same mistake twice, but they make most mistakes once.

Searching Algorithms

The value-based decisionmaking methods discussed previously require a nearly exhaustive search of the set of decisions to be effective. Some of these algorithms guide the search with enlightened decision rules or limit the search by pruning off bad decisions. When the space of possible decisions becomes large, still more-efficient searching techniques must be used. One such technique is the localized search.

If the set of decisions can be partitioned into subsets such that any two subsets differ significantly but decisions in the same subset share a high degree of commonality, the efficacy of the search increases. A representative decision from each subset is first evaluated, then the subset whose representative was judged best is selected for further examination. This procedure may be applied again to the subset to reduce further the number of candidates that need to be examined. The key is to keep the required representative evaluation manageable at each level of partitioning.

If the decision to be made is hierarchical, the partitioning is straightforward. The partition is based upon the decision at each level of the hierarchy. For example, the various theater-level decisions may be partitioned; within those, the corps-level decisions; within each of those, the division-level decision; and so on. Using the COFM methodology of the C^2 Planner previously discussed, first the theater commander allocates his reserves to the corps. Then, the corps commander parcels out his assets to the divisions. The process continues until the decision for the lowest-level command has been made.

Objective-Driven Optimization and Gaming Methods

Another procedure for efficiently searching the decision space is to guide the search with the objective of maximizing the value of the chosen decision. The function that determines the value of the decision is called the objective function, and the principles of mathematical programming are brought to bear to maximize the objective function efficiently. Some of the methods used involve linear programming and shortest path and maximum flow algorithms.

The use of an objective function and mathematical programming solution techniques allows a further improvement in the representation of decisionmaking. In combat, an important aspect of decisionmaking is its two-sided nature. Based upon the perceived objectives of the opposing side, the decisions from earlier periods, and the results of earlier interactions, each side adapts, in effect, to the decisions of the other side. To take advantage of the two-sided nature of decisionmaking in combat, the objective function should also be two-sided, increasing as the outcome favors one side and decreasing as it favors the other.[3] The goal of the procedure then becomes the simultaneous maximization of the objective function with respect to one side's decisions and the minimization of the objective function with respect to the others side's decisions. The decision space is further refined by the introduction of constraints on the decisions and actions of each side. In mathematical programming parlance, this is known as a competitive game. The solution decision derived for a game is usually not the optimal or best for either side but is a saddle point that balances the value of the outcome for each.

Construction of a game's objective function helps to facilitate representation of the interactive dynamics of combat. TLC uses the Sequential Analytic Game Evaluator (SAGE) algorithm to derive the solution to a game.[4] Suppose a new and more capable weapon is introduced by one side. Given that the other side knows about it, that side will tend to adapt its use of existing systems to neutralize the new system's effect as much as possible. The use and effectiveness of the new system will be different from what they would be if the opponent's response did not occur. This can also force a change in the behavior of the side that introduced the new system, reducing the marginal contribution of the new system.

[3]It is quite possible that the objectives are not mirror images of each other, however. In this case, there is the possibility of "cooperative" solutions in which one side can ignore or help elements of the opponent's objective to do better on its own objective.

[4]SAGE is partially described in Appendix B.

The example shown in Figure 6.1 is from a study of air defenses in NATO's central region in the 1980s and illustrates the value of an objective-driven adaptive game for the allocation of key resources in a simulation.

As the figure shows, two cases are simulated, a base case and one with additional Patriot surface-to-air batteries in the theater. The left-hand graph shows an apparently counterintuitive result. As the number of surface-to-air systems increased, the number of enemy aircraft that the surface-to-air defenses shot down actually decreased. The right-hand bar chart shows, however, that the effect of additional batteries was to reduce the territory NATO lost. The objective metric for the example was the penetration by the Warsaw Pact into NATO territory.

What actually happened in the simulation was that the Warsaw Pact commander, represented by an objective-driven adaptive decision algorithm, adjusted his decisions when the Patriots were increased, by shifting to a strategy of penetrating with fewer aircraft and in areas not protected by Patriots. In effect, the Soviet air planner was "deterred" by this additional NATO capability. This does not occur in models with scripted or nonadaptive planners; they would, instead, show increasing and decreasing Red losses as a function of increasing and decreasing numbers of surface-to-air defenses, because the plan would remain unchanged. Also, this adaptation does not occur in higher-resolution mission or penetration modeling, in which the force allocations are inputs and the higher-level command and control systems relating to force allocation are not present. Adaptive planning within the campaign model is very important for evaluating resource effectiveness in a combat environment. In

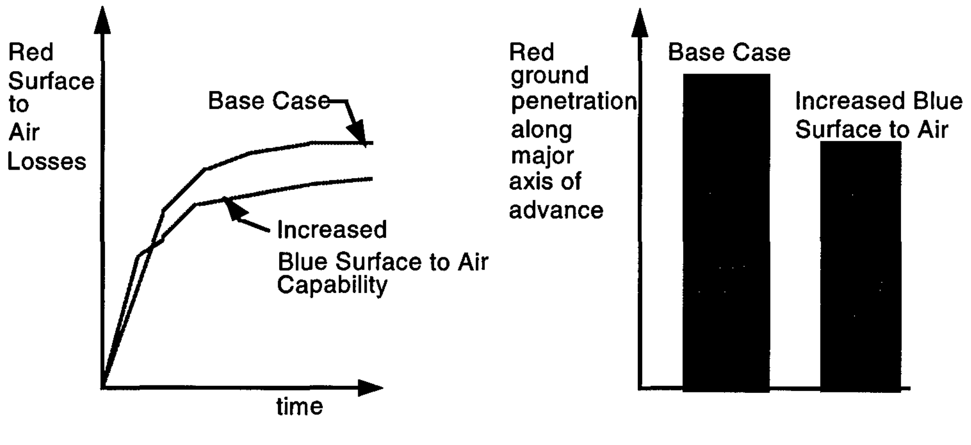


Figure 6.1—Objective-Driven Adaptive Simulation: Patriot Example

terms of policy analysis, the adaptive approach to strategy formulation can lead to better understanding of these sorts of dynamics; because it is objective driven, the payoffs may be different (e.g., for deterrence as opposed to for attrition) from those a static analysis might have suggested.

Allocation of Flexible Assets with the SAGE Algorithm

SAGE[5] is used to optimize allocation of air and other flexible combat assets, such as missiles and helicopters, to meet theater objectives in TLC.[6] We use the word "allocation" for the general process of determining a multistage (multiday) plan of tactical employment within a planning command that addresses the time, geography, and function of employment.[7] A good or "best" employment strategy depends on the objective, which may be territorial, attrition related, or strategic. It also depends on the opponent's strategy, which is where analytical gaming plays a role. Finally, it depends on the capabilities and force structure of each side and on the employment constraints (political constraints on border crossing, for example). The key features of the SAGE algorithm deal with each part of the tactical employment problem. In this subsection, although we might use long-range fires, helicopters, or other resources to make our points, we will illustrate the allocation of flexible assets by focusing on air employment.

SAGE consists of an algorithm placed within the TLC multiperiod conflict simulation that allows user specification of overall objectives that are to be minimized (e.g., own attrition) or maximized (e.g., attrition of opposing forces) and engages in an automated search for the "best" strategies to meet the user-specified objectives. With the SAGE algorithm, the user specifies a measure of merit (e.g., targets or resources destroyed, final position of the forces, force ratio at the end of the conflict), and the methodology specifies a sequence of strategies for the allocation of resources of the two opposing sides.[8]

[5]The mathematics of SAGE are described in Appendix B.

[6]As with the ground C^2 planner, scripting is also allowed.

[7]The components of tactical air employment include *apportionment*, or the level of effort to each mission type such as air defense; offensive counterair (attacks against air bases); and ground support (attacks against opposing ground forces). Air employment also includes *allotment* of aircraft between air planning commands, the *allocation* of aircraft—which translates the apportionment or level of effort into sorties by type of aircraft—and the *tasking* or assignment of specific flights to specific targets.

[8]SAGE solves the multistage problem by what amounts to a piecewise linearization of the optimal value function of dynamic programming. The SAGE algorithm arrives at the linearization point by an iterative, fixed-point approach and does not require the combinatorial evaluation of optimization problems at each stage as a function of a large number of states. See Appendix B for a more complete description of the algorithmic approach.

SAGE solutions indicate the strategy that maximizes the marginal payoff of the attack aircraft, whether they attack land forces, air bases, or other targets. SAGE is used in TLC to allocate aircraft, long-range fires, and helicopters to mission types and interacts with the ground-force operational planner described earlier. It is also to be used to provide situation-dependent target values for other algorithms. Figure 6.2 illustrates the operation of SAGE within the TLC simulation.

The SAGE algorithm is used to represent each planning command, with the possibility of more than one planning command—independently maximizing different objectives—on each side. Furthermore, each side has its own perception of the status of its own forces and situation and a perception of the other side's capabilities. Within each of the planning commands, SAGE is used to run simple, "look-ahead" simulations to play out the consequences of various allocations of resources; in this regard, SAGE assists in gaming the next several periods to better assess the current operational plan. Each side's look-ahead is essentially a conjecture on what the current model of the world is and on what the consequences are of pursuing its objectives while the opposing side pursues its *perceived* objectives. At the conclusion of each planning cycle, reallocation again takes place based upon the look-ahead simulation. Figure 6.3 illustrates the use of perceived information and simple look-ahead models in the TLC implementation of SAGE.

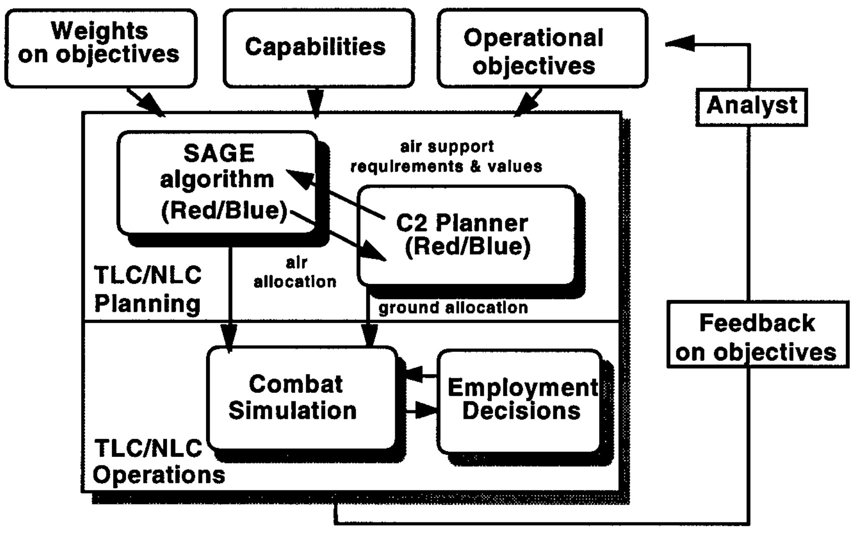


Figure 6.2—SAGE Algorithmic Process

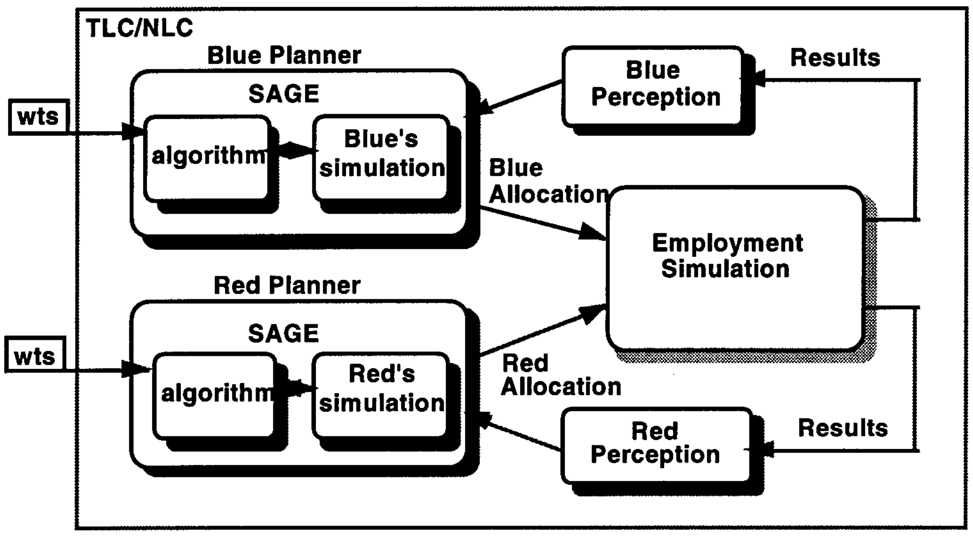


Figure 6.3—SAGE Implementation in TLC

Each planning command may have a number of air bases and units at its disposal, as well as a number of possible targets and ground units to support.

The process by which SAGE allocation takes place at the aggregate level has three steps:

- *Apportionment*—In the first step, SAGE is used for apportionment—i.e., identifying the level of emphasis (the proportion of sorties) appropriate for different air efforts (air defense, offensive counterair, etc.). The required level of effort is addressed by apportioning sortie requirements—attempting to allocate the best type of aircraft and the best type of unit (i.e., by training level) to particular missions.
- *Allocation to Targets*—In the second step, units and their sorties are matched to possible targets on the basis of the value of the targets, part of which derives from the objectives of the apportionment process; this is called *target-type valuation*. Given a target-type value and a unit with a particular type of aircraft, the amount of value from attacking the target is based on the value of the target. The algorithm seeks to find the optimal assignment of sorties and units against the best set of targets, given the objectives.
- *Assignment to Time Periods*—In the final step, the sorties and units are assigned to discrete time increments that each day comprises, with sorties launched during their assigned increment of time. Sorties are generally distributed uniformly as aircraft become available, except that some may be specifically assigned to day or night missions. The targets with the highest

time urgency are first assigned sorties; targets that have to be dealt with sequentially (e.g., air defense systems) also must be prioritized. Some assets will also be allocated on a support basis. (For example, if air attrition is high, the algorithm may allocate assets to defense-suppression missions.)

The end result of this process is a number of air tasking orders (ATOs) to specific air units.[9] ATOs—whether created by input or generated algorithmically—include all necessary flight path information for the aircraft to reach its target via the network.

The SAGE Algorithm Illustrated

The mathematical description of the SAGE algorithm is provided in Appendix B. In this subsection we will first discuss the steps of the process and then illustrate it applied to a simple example. Figure 6.4 shows the basic steps of the process.

The algorithm performs a search process that uses a simplified simulation model of Red and Blue, internal to the TLC simulation, to develop a campaign plan. This campaign plan is generated by repeatedly "playing" the simplified simulation over complete campaigns with alternative allocations of resources for each side until an equilibrium point is reached. As seen in the figure, the process

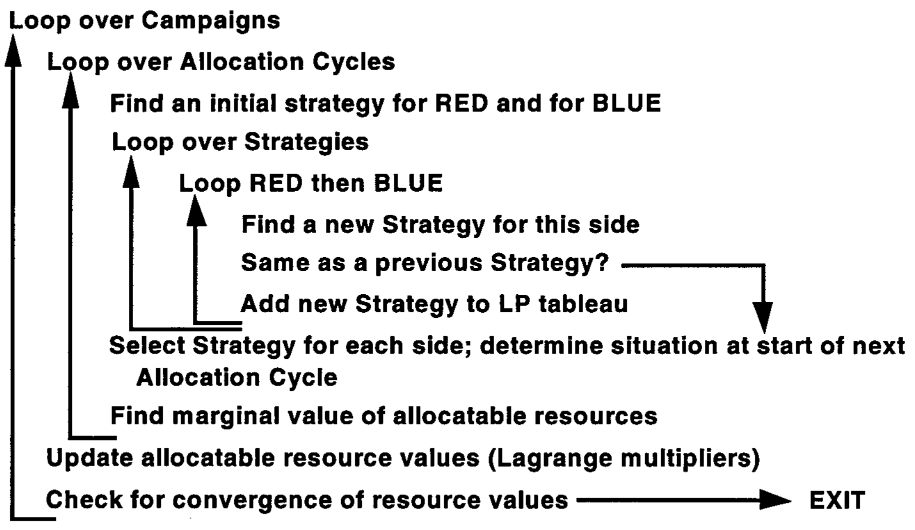

Loop over Campaigns
Loop over Allocation Cycles
Find an initial strategy for RED and for BLUE
Loop over Strategies
Loop RED then BLUE
Find a new Strategy for this side
Same as a previous Strategy?
Add new Strategy to LP tableau
Select Strategy for each side; determine situation at start of next Allocation Cycle
Find marginal value of allocatable resources
Update allocatable resource values (Lagrange multipliers)
Check for convergence of resource values
EXIT

Figure 6.4—Steps of the SAGE Algorithm

[9]The model also allows the user to script ATOs.

loops over multiple campaigns, tries alternative allocations in each allocation cycle, cycles over Red and Blue, determines marginal values for the resources on each side, and continues this process until it converges on the best equilibrium strategies of each side. The marginal values of the resources, developed by examining how the resources affected a previous campaign in the cycle, provide the means to determine how to make the best decision on any particular allocation cycle without having to make decisions in all subsequent cycles simultaneously. These marginal values or Lagrange multipliers indicate,[10] for instance, what an aircraft might be worth for the remainder of the campaign in terms of the objective. Thus, an aircraft might be worth 10 combat vehicles for the remainder of the campaign if the objective were to destroy combat vehicles. With such values, the algorithm can then determine whether it is more beneficial to destroy an aircraft of the other side or to attack combat vehicles directly during a particular allocation period.

The SAGE process uses gradient search techniques to vary the allocation strategy to improve the objective value of one side at a time. These techniques adjust the allocation of aircraft to missions a little at a time and sense the effect on the campaign in the simplified campaign model. Once a direction of improvement is found, the algorithm changes the allocation in that direction until it is no longer improving, because there is nothing more to allocate. Figure 6.5 lists the steps of this process.

The initialization of this search deserves some comment. Because the search is over a generally nonlinear objective function, it is possible for hill-climbing methods to get "stuck" in a local or relative optimum, not finding the overall best solution. This can happen, for instance, in deciding whether or not to perform a mission penetrating enemy defenses. If only a few aircraft are allocated in the perturbation process, they may all get shot down by the enemy defenses, and the mission may not look promising. However, if more aircraft are sent, the defenses may be penetrated, and a high value target may be destroyed. We handle this by starting each aircraft at its best mission and then backing off that mission in relatively big steps at first and then smaller steps to refine the solution.

The easiest way to see how the SAGE method works is to illustrate it with an example. The following example is described mathematically in Appendix B. Here, we will describe it in nonmathematical terms. It is highly simplified and created only as an illustration. The next subsection will say more about the

[10]In mathematical programming, the Lagrange multipliers are the so-called "shadow prices" of resources, which indicate how much better the solution would be if one more resource was provided.

```
For each Allocatable Resource on this side,
    Find best Mission for full Allocation, when nothing else on this side
      flies (best Mission may be stay at home)
Set Strategy to full Allocation to the best Mission for each Resource
Loop over gradient searches
    For each Mission of each Allocatable Resource on this side,
        Find effect on Objective of perturbing Allocation to this Mission
    Select perturbation direction with largest Objective gradient
    Change Allocations along this direction until no further improvement
      in Objective
    Same as previous Allocations? ──> EXIT
    Set Strategy to the new set of Allocations
```

Figure 6.5—Steps in the SAGE Search Process

complexity of the internal SAGE simulation. Figure 6.6 gives the elements of this example.

The objective of each side is to maximize its net ground support over a four-day campaign. The effectiveness of each side is the same in the ground-attack mission, and the only other mission is to attack the aircraft of the opponent at his air bases. In this air base attack mission, the Blue force has a slight advantage in

Objective:

- **Blue - maximize net ground support**
  (wb.Blue Ground Sorties - wr.Red Ground Sorties)
- **Red - minimize the same measure**

Relative Capabilities:

- **100 aircraft on each side initially**
- **1 aircraft type on each side, sortie rate 1.0, two possible missions**
  —Ground Support (GS) and Airbase Attack (ABA)
- **GS effectiveness for both sides = wb = wr = 10**
- **ABA effectiveness - Blue destroys 0.6 aircraft per sortie**
  - Red destroys 0.4 aircraft per sortie

Problem:

- **Over 4 days, what mix of missions should be flown?**

Figure 6.6—Statement of a Simple Example to Illustrate the SAGE Algorithm

that each aircraft sortie of his is expected to destroy 0.6 of a Red aircraft (a kill probability of 0.6), and each Red sortie in air base attack is expected to destroy 0.4 of a Blue aircraft. The problem is how Blue and Red should allocate their 100 aircraft among these two missions.[11] Table 6.2 is a table showing the first step in the process.

Each row of the table shows the allocation of both sides' aircraft to the two different missions on one day of the campaign. Initially, the aircraft are all assumed to have a zero marginal worth; thus, allocation to air base attack, which destroys airplanes, provides no worth to the overall objective. On the other hand, attacking ground forces gives each side 1,000 units each day (the effectiveness of 10 times the 100 aircraft allocated each day). The net value of these allocations is 0 each day because each side gets the same value. In effect, Blue gets 1,000 points for attacking Red's ground forces, but he loses 1,000 points because of Red's attacks. At the end of the first iteration of the campaign, the aircraft are revalued. The marginal worth of the aircraft is computed as the value each aircraft would contribute to the remainder of the campaign if it survived that day. Thus, at the end of day 1, a Blue aircraft is worth 10 points × 3 days, or 30 points. At the end of day 2, it is worth 20 points; at the end of 3 days, it is worth 10; and it is worth nothing at the end of the campaign.[12] Red aircraft are similarly valued. This is shown in Table 6.3.

Table 6.2

Iteration 1 of SAGE Example

SAGE Iteration 1

	A/C Value		Blue		Red		Net	A/C Left	
Day	Blue	Red	GS	ABA	GS	ABA	GS Value	Blue	Red
1	0	0	1	0	1	0	0 *	100	100
2	0	0	1	0	1	0	0	100	100
3	0	0	1	0	1	0	0	100	100
4	0	0	1	0	1	0	0	100	100
					Total Net GS Value		0		

*** 100 Blue GS Sorties x wb - 100 Red GS Sorties x wr = 1000 - 1000 = 0**

[11]This problem was first described in the original TAC CONTENDER work (U.S. Air Force, 1971).

[12]The actual algorithm allows resources to have a residual value at the end of a campaign so that there is a reason to live to fight another war.

Table 6.3

Adjusting the Marginal Values in SAGE

SAGE Iteration 1+

	A/C Value		Blue		Red		Net	A/C Left	
Day	Blue	Red	GS	ABA	GS	ABA	GS Value	Blue	Red
1	30	30	1	0	1	0	0	100	100
2	20	20	1	0	1	0	0	100	100
3	10	10	1	0	1	0	0	100	100
4	0	0	1	0	1	0	0	100	100
					Total Net GS Value		0		

Blue A/C Value at end of day 1 =
3 sorties x wb = 30

At this point, the algorithm makes a new attempt to allocate aircraft in the campaign. Now, using the aircraft marginal values, it can determine that, if a Blue aircraft attacks air bases on day 1, it will generate 30 × 0.6 = 18 units of value, instead of only 10 units of value if it attacks ground forces. A Red aircraft will obtain 30 × 0.4 = 12 units, instead of the 10 units from ground attack. Thus, both sides allocate all of their aircraft to air base attack on the first day. The results of these attacks are 60 Red aircraft destroyed and 40 blue aircraft destroyed. This is shown in the first row of Table 6.4.

On the second day of the campaign, the Blue evaluation shows that air base attack gives 20 × 0.6 = 12 units of value per aircraft, and ground attack gives 10

Table 6.4

The Second Iteration of SAGE

SAGE Iteration 2

	A/C Value		Blue		Red		Net	A/C Left	
Day	Blue	Red	GS	ABA	GS	ABA	GS Value	Blue	Red
1	30	30	0	1	0	1	0	60	40
2	20	20	0	1	1	0	- 400	60	4
3	10	10	1	0	1	0	560	60	4
4	0	0	1	0	1	0	560	60	4
					Total Net GS Value		720		

units. Thus, the decision for Blue is to continue air base attack with all of its remaining 60 aircraft. For Red, the evaluation shows that air base attack provides $20 \times 0.4 = 8$ units instead of 10 units for ground attack, and the decision is to devote its remaining 40 aircraft to ground-unit attack. This is shown in the second row of Table 6.4. On the third day, Blue's evaluation shows that air base attack is now worth only $10 \times 0.6 = 6$ versus the 10 for ground support so Blue decides to switch its 60 aircraft to ground support. Red's remaining four aircraft continue to perform ground-support attacks. This is shown in row 3. The evaluation in the fourth period leads to the same allocation, as shown in row 4. Note how the declining marginal worth of the aircraft as the campaign progresses, computed from the first iteration, guides the allocation in this subsequent iteration. These values provide the information needed to make daily term-planning decisions that are optimal for the entire campaign without actually looking ahead at each day. The look ahead has been provided by running the campaign multiple times and deriving the aircraft's marginal worth from the previous campaigns. This is the key to the SAGE algorithm. One more iteration would be required for this example to verify that it has converged. The reader can verify himself that a recomputation of the marginal worth of the aircraft would give a slightly higher value for the Blue aircraft based on this campaign but that the allocations would remain the same.

It is important to note that this was a highly simplified example and that, in reality, many more aircraft missions would be represented; the interaction model would be more complex; other resources besides aircraft might be allocated; and many more iterations are likely to be required for convergence. Furthermore, the solution is likely to require partial allocations of the resources to several missions, rather than all or nothing, and a "mixed strategy" solution may be required. The mixed strategy occurs because there may be no stable equilibrium in which one side can announce its allocation, let the other side optimize, and still expect its original strategy to be best. This is actually the more likely case; to solve it, the SAGE algorithm develops a probabilistic mix of strategies in which the probabilities are known to each side, but the actual allocations to be used are not. This is the standard game-theoretic approach to finding a game equilibrium.

The Combat Simulation Internal to SAGE

There are two ways of implementing the SAGE algorithm in a combat model. One could place the algorithm around the TLC simulation and develop the resource allocations of each side by iterating the main simulation and replaying many campaigns. However, in developing the allocations, SAGE may play each day of a campaign 1,000 or more times and may replay the campaign hundreds of times. The time penalty for doing this around a simulation of the complexity of TLC would be excessive. Furthermore, the algorithm would then be operating with perfect information about effectiveness and resources of each side, and the objectives of each side would necessarily be symmetric, Blue's objective being the negative of Red's. The second alternative is to place SAGE internal to the main simulation and use a second, more aggregate combat model internal to the SAGE algorithm. Then, each side can use its own representation of the battle, objectives, and perception of the other side. This is how SAGE was implemented in TLC, and this is what was shown in Figure 6.3. In fact, when there are multiple planners on each side, multiple SAGE algorithms can be implemented for a side to represent the allocation of resources under each planner's control. This subsection provides a brief description of the internal combat simulation of SAGE as implemented within TLC.

The internal simulation is an aggregate version of TLC's actual simulation. Remember that the purpose of this internal simulation is to permit the SAGE algorithm to develop resource allocations that will be used by each TLC air-planning command. The targets, missions, aircraft, and combat interactions are simplified for quick running but contain enough information to permit good resource allocations to be developed. The allocations for a planning horizon are provided to TLC, but only the allocation for the current planning period is actually used. At the next TLC planning period, the SAGE algorithm moves the planning horizon ahead and recomputes the allocations. Each time this occurs, the internal combat simulation of SAGE is updated with information about the state of resources and the achieved results from the previous planning period. This sequence of events is depicted in Figure 6.7.

The battle space is divided into sectors or regions to differentiate ranges between air bases and targets and possible differences in terrain or air defenses. The range between all air bases in one region and all targets in another is assumed tobe the same. The targets and regions included in the aggregate simulation are depicted in Figure 6.8.

The missions allowed in the aggregate simulation are depicted in Figure 6.9. These missions are allocated to the various aircraft types and missiles, are flown against the targets shown in Figure 6.8, and are allocated by sector.

The internal combat simulation runs in time steps, advancing through a day in a sequence of periods called sortie cycles at which all sorties for each day are generated. The steps in each period are described in Figure 6.10. Each day is simulated exactly the same by SAGE from the current planning period to the end

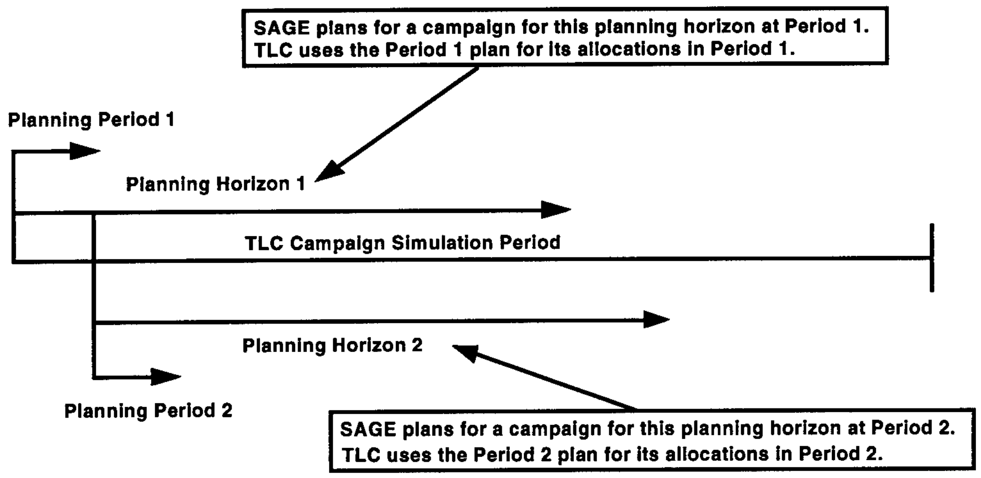

SAGE plans for a campaign for this planning horizon at Period 1.
TLC uses the Period 1 plan for its allocations in Period 1.
Planning Period 1
Planning Horizon 1
TLC Campaign Simulation Period
Planning Horizon 2
Planning Period 2
SAGE plans for a campaign for this planning horizon at Period 2.
TLC uses the Period 2 plan for its allocations in Period 2.

Figure 6.7—Interaction of SAGE Planning with TLC

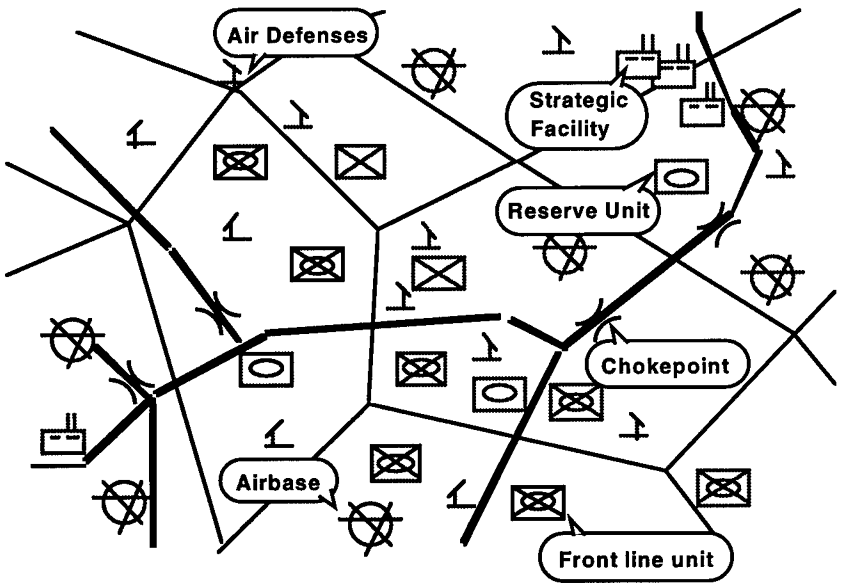


Figure 6.8—Targets in the SAGE Internal Combat Simulation

of the planning horizon. At the end of each day, the results of that day's activity are used to update the state of resources for the next period's plan. SAGE plays (simulates) each day or planning period multiple times as it attempts to find the equilibrium allocation for each side of the internal simulation.

At each cycle during the day, missiles are first allocated to targets, and those allocations are simulated. After the results of those firings are determined (air bases may be closed, for example), the aircraft are allocated to missions, and

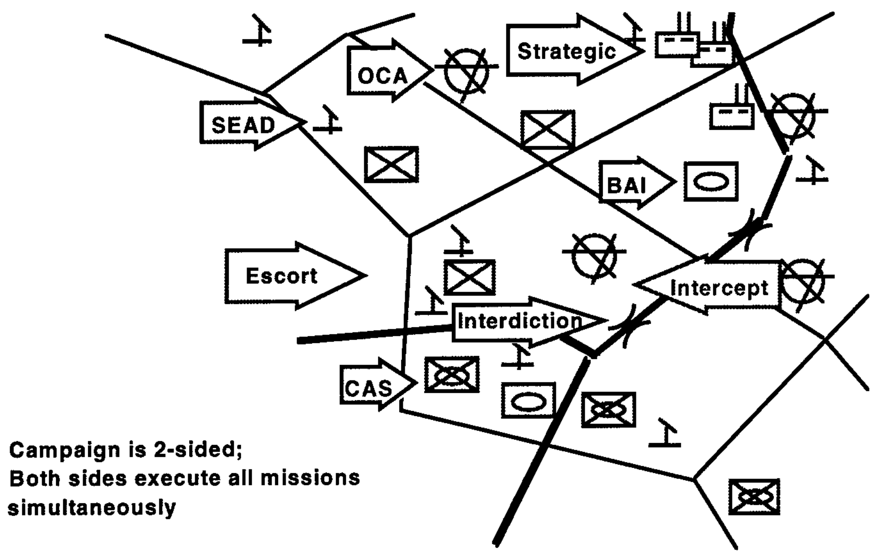


Figure 6.9—Missions in the SAGE Internal Combat Simulation

- **Set initial Resource Numbers and Air Base states**
- **Add Reinforcements**
- **Execute a number of Sortie Cycles:**
  - **Allocate Missiles to Targets**
  - **Execute Missile Firings and apply results**
  - **Allocate Aircraft to Missions and Targets**
  - **Execute Aircraft Sorties and apply results**
  - **Determine Airbase and Aircraft availability for next cycle**
- **Record state at end of day**
- **Calculate Game Value**

Figure 6.10—A Day in the SAGE Aggregate Simulation

sorties are generated. The results of aircraft arriving at targets from previously generated sorties and air base availability are then determined. This process continues throughout the day (planning period) with both sides executing at the same time in each cycle. The missile-firing events are depicted in Figure 6.11.

Figure 6.12 depicts the aircraft mission activity in each cycle. Aircraft are subject to attrition by surface-to-air and air-to-air defenses in each region, as well as

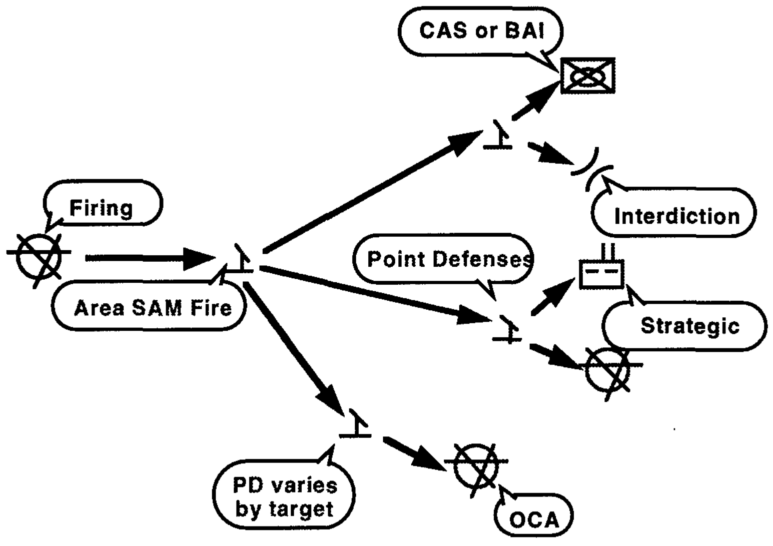


Figure 6.11—Missile Firing Events in the SAGE Internal Simulation

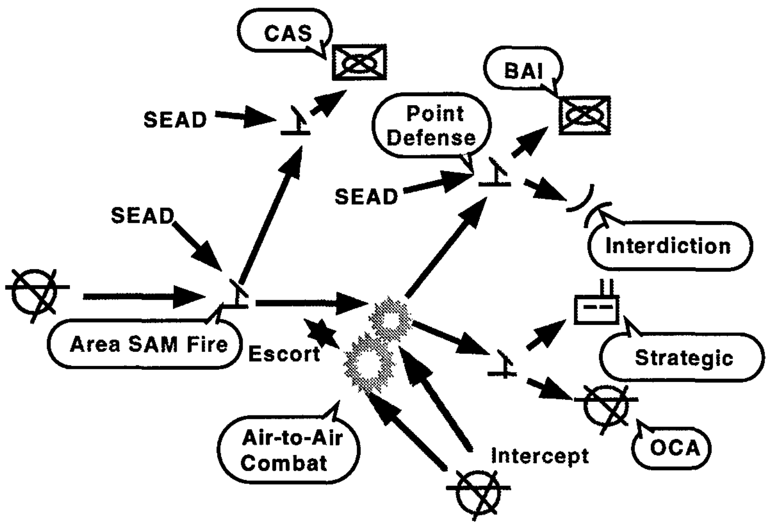


Figure 6.12—Aircraft Activity During a Cycle of the SAGE Internal Simulation

point-defense attrition at a target. SEAD activity can suppress the surface-to-air defenses, and escort missions can be used to protect the penetrating attackers. The targets in each region are those depicted earlier. Although this internal simulation is aggregate in time, space, and resources, it is rich enough to capture the important aspects of the battle to provide a reasonable allocation to the TLC planner.

Illustration of the SAGE Allocation of Aircraft Missions

The example in this subsection illustrates how SAGE adapts an aircraft allocation based on the situation at hand. This example uses the actual internal simulation within the TLC implementation of SAGE. A number of aircraft types are available to the planning commands, and the missions and targets described earlier are represented. In the base case for this example, the ground units are judged by the algorithm's prescribed objective to be the most valuable targets; so with the exception of SEAD and escort, the aircraft are allocated to the ground support missions, CAS, and BAI for the 5 days of the planning horizon, as shown in Figure 6.13. The results in terms of Red ground targets destroyed by these sorties are also shown in the figure.

In the next case, the ground targets were judged to be unavailable as targets during the first three days of the campaign. At the first iteration of the algorithm, it chooses to allocate the ground-attack sorties to strategic targets as an alternative. This allocation and the resulting effects on target value destroyed are shown in Figure 6.14. The strategic targets, while valuable, were also heavily protected by air defenses, and the resulting aircraft attrition leads to the drawdown in the Blue force shown in the left hand chart of Figure 6.14. The allocation switches to ground-unit attack after day 3. Note that the value obtained in this iteration is well below the base case because of the lack of lucrative targets, and the aircraft attrition is higher as a result of attempting to attack the heavily defended targets.

The final iteration of SAGE achieves a different allocation of the Blue aircraft. As shown in Figure 6.15, the aircraft are actually held back until the ground unit targets become available. Offensive counterair (air base attack) missions are

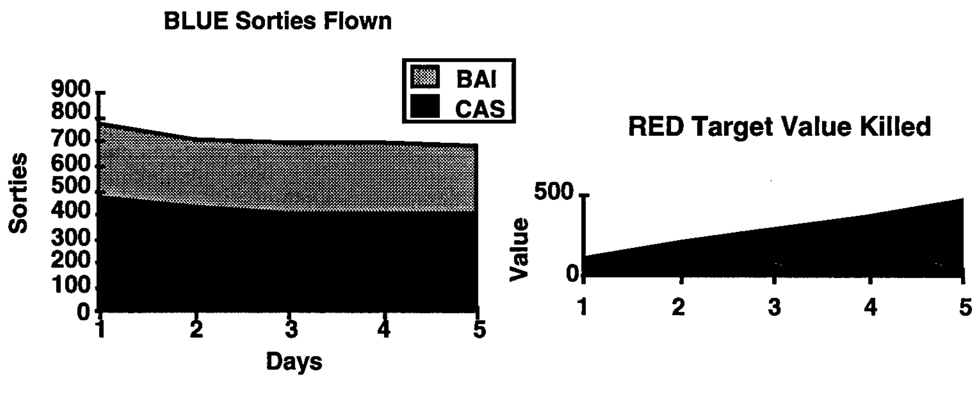


Figure 6.13—Base Case Allocation—Ground Units Available as Targets

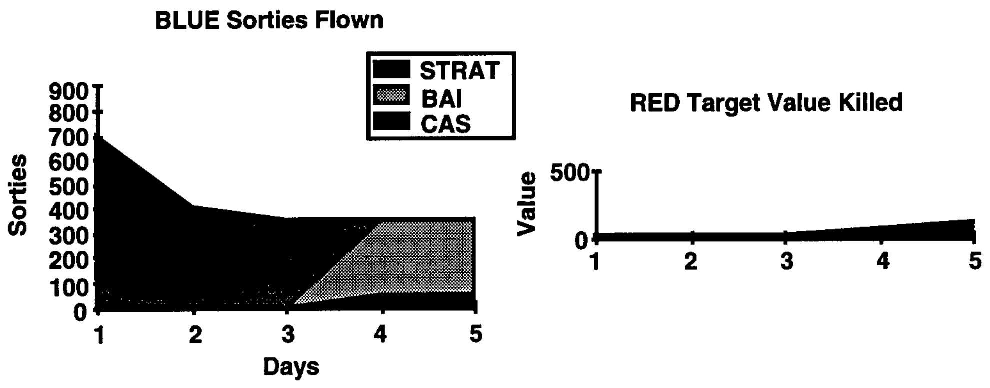


Figure 6.14—First Iteration—Altered Target Availability

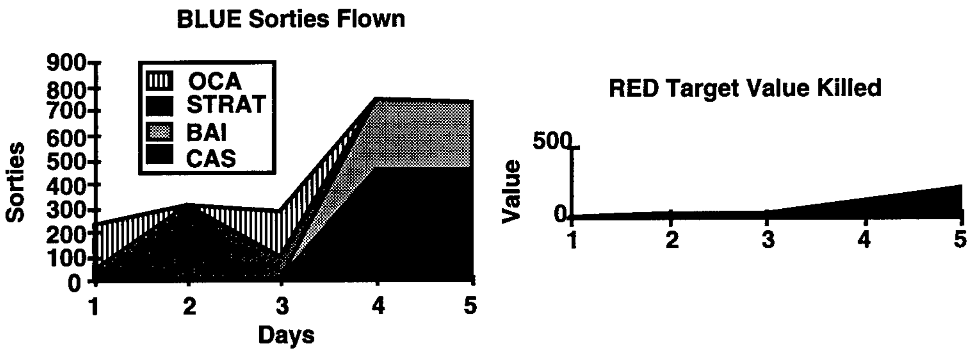


Figure 6.15—Final Iteration—Altered Target Availability

flown, and fewer strategic targets are assigned. The results of this are that more Blue aircraft survive to attack the ground targets when they are available and that the net Red target value destroyed is somewhat higher, as shown in the right-hand graph of the figure. The SAGE algorithm, acting on predicted target availability from TLC, chose to withhold aircraft from immediate and expensive attacks and to perform the air base attacks to reduce the defenses. It was then able to be more effective in the later attacks, when the targets were predicted to become available. Of course, the goodness of this campaign is a function of the accuracy of the prediction of target availability, but that is what the simulation and measurement of intelligence capability is about.

The use of an automated planner reduces the burden on an analyst to change the details of the force allocation whenever scenario, force structure, or effectiveness changes are to be examined in the simulation.

User Control of the SAGE Algorithm

SAGE was developed to provide a complete optimization in the determination of resource allocations in a simulation. The user is provided a number of ways of controlling this process with respect to objectives, forcing specific allocations, changing the planning horizon, and constraining the time required by the iterative process. The objectives are controlled by the weights attached to strategic targets as opposed to support of the operational battle. Heavy weighting of the strategic targets will skew the aircraft allocations toward those targets relative to those more directly linked to the operational conflict. Resources are also given end-state values, so that placing a heavy value on aircraft will cause the allocation to be relatively careful about assigning missions that would lead to heavy losses. Thus, one can force a relative concern about attrition of key assets. The planning horizon and frequency of replanning are input parameters, so that planning can be done with little concern for the future or with a heavy emphasis on the long-term effects of immediate activities. Allocations can be suggested or forced upon the SAGE algorithm. Input allocations can be tagged as candidates for the algorithm at particular times, in which case SAGE will start with those inputs and then attempt to find something better. Or allocations can be tagged as "must do," in which case they are the only allocations considered during the specified time period. Other details about resource assignment can also be input to the algorithm. Specific targets can be assigned, and specific missions can be predefined. The implementation of the algorithm in TLC then works around these assignments, using them first and then assigning the remaining resources or targets.

Control of the execution time of the algorithm can be accomplished in several ways. The frequency of calling SAGE can be controlled so that, for instance, it can be called every x planning periods to reoptimize, and TLC will use the allocations of resources developed earlier to fill in. TLC then assigns new targets and resources based on availability, while following the prescribed overall allocation. Another control is in the number of iterations allowed to find solutions at either the campaign or planning-period level, or between the air and ground planners. The fact that the algorithm is operating on imperfect, perceived information and is only suggesting an allocation to TLC, which adjusts the plan to fit the specific resources and targets currently available, allows some compromises to be made in the search for optimality while still achieving reasonably good resource plans.

Interactions Between Ground and Air Planning in TLC

The interaction of the C^2 Planner and the SAGE air planner is depicted in Figure 6.16. For each attack in the ground forces' operational plan, the C^2 Planner develops a *support request list* of ground-unit targets for each side and a value for that target set based upon its contribution to the substantiation of the attack.[13] These values are used in the SAGE algorithm to develop a *support tasking list* of ground-unit targets and the amount of effort to be placed against each. These taskings are evaluated by the C^2 Planner; if they are insufficient, the C^2 Planner will redevelop the support request list, perhaps with new values. The procedure iterates until sufficient support taskings are produced or until the C^2 Planner concludes additional effort from SAGE is not forthcoming. This process means that there are additional iterations of the SAGE algorithm and the C^2 Planner as they converge to a solution. Practically, we have found that two or three cycles of this iteration usually achieve a stable set of ground-support sorties. The iteration process starts with the ground forces' C^2 Planner, which initially assumes that the ground-support sorties in the current planning period will be similar to those provided in the previous planning period. The SAGE algorithm then uses the proposed targets in the support request list and may modify the support to be provided based on other competing targets, updated information about enemy defenses, etc. The result of this process is a simulation of the air-

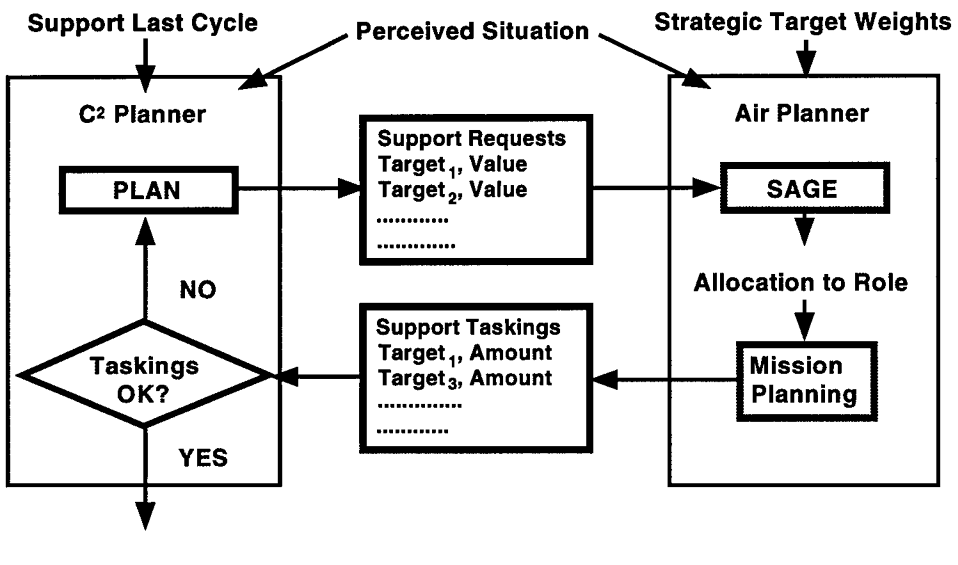


Figure 6.16—Ground and Air Planning Interact

[13]See Section 5 for a description of the ground forces' operational plan and the C^2 planner algorithm.

support planning and the negotiation between ground commanders and the air planner.

Observations About Adaptive Planning and the SAGE Algorithm

TLC's combination of optimizing algorithms, scripting rules, and decision tables contributes to the treatment of operational strategy as an iterative and adaptive process, while enabling the user to retain control over the objectives and the extent of algorithmic optimization in the execution of an operational plan. We will make a number of observation here for those interested in implementing such algorithms in a simulation.

The Importance of Objective-Based Adaptive Planning

We have shown by example earlier in this section how allocations in a campaign model differ when they are allowed to adapt to the capabilities at hand. Allowing such adaptation may seem contrary to the scientific approach, which attempts to hold as many variables constant at one time as possible to distinguish cause and effect. Unfortunately, for this type of analysis, the capabilities of a military force are strongly intertwined with the way the force is used. It would be foolish, for example, to develop a stealth aircraft and then require it to penetrate defenses in the same way as a nonstealthy aircraft, requiring the same level of defense suppression, etc. Similarly, a weapon with a greater range should take advantage of that range rather than being used in the same way as its shorter-range predecessor. There are several important aspects of understanding how the use of force might change as the capabilities of either side change. An opponent's strategy adaptation to a new capability of its foe may partially defeat the purpose or effect of that capability. Thus, weapon systems may not achieve the value expected without such an adaptive environment. Mechanisms for the dispersion of submunitions from a launched dispenser might be defeated by alternative approaches to movement of ground units, for example. Although the enemy may not always adapt because he either chooses not to or does not immediately know how to, it is nevertheless important to understand how he *could* adapt.

A second aspect of adaptation is one's own adjustment to capabilities. Frequently, new tactics or strategies are dictated by the introduction of new capabilities. There are numerous historical examples of how warfare changed as the result of the introduction of a new capability. These include operational concepts wrought by the machine gun, the tank, and the aircraft carrier.

Although models have strong limits on their ability to invent new tactics, a model that adapts strategies to capabilities can sometimes help the attempt to understand the influence of capabilities on operational concepts and tactics.

The relative capabilities of each opponent naturally lead to asymmetric strategies, and it is important to have adaptive mechanisms to understand these. In our own work with SAGE during the Cold War, we often found that doctrine did not match the best use of force against the capabilities of the threat that was faced. Heavy focus on air base targets would lead to excessive attrition with very limited suppression of enemy aircraft, etc.

Earlier in this section, we discussed a number of alternatives to adaptive planning, including decision tables and other algorithms. One problem with most of these approaches is that the objective is often not apparent. Certainly, there was an implicit objective as the rules and algorithms were developed, but this is not apparent, and it is therefore not obvious or easy to change the rules when there is a different objective. In contrast, in an algorithm, such as SAGE, the objectives of each side are made explicit and can be changed. Thus, one quickly sees how the emphasis on targets, attrition, movement, etc., can change the nature of the solution.

The Importance of Modeling Decisionmaking Under Uncertainty for the Analysis of Command and Control

We believe that a key to understanding the value of command and control systems is to understand the decisionmaking and the uncertainties that those systems reduce in support of decisionmaking. Many current campaign models use nearly perfect intelligence in making force-allocation decisions. This makes it very difficult to evaluate new systems that attempt to improve the quality or timeliness of information. There is simply nothing to improve, and the hypothesized force multiplier effect does not show up. Furthermore, if the decisionmaking does not adapt to new and better information, such as is the case in scripted models, there will also be no value shown for better C^4I. In the TLC research, we tested one approach to placing an adaptive algorithm *within* a simulation so that a perceived view of the enemy could be used in the planning process. Because of the adaptive decisionmaking, the more correct this view is, the better the plan. The perception should include not only forces but the estimated objectives and capabilities of the other side. This approach permits a model of imperfect intelligence and the fusion process to be placed within the model's decision processes. We do not claim to have modeled the intelligence-fusion process all that well in this research, but the mechanisms for utilizing and adapting to fused intelligence have been created and demonstrated in TLC.

The Need to Understand the Algorithms Used for Adaptive Planning

The advantages of the adaptive-planning approach described in this section also come with a price. The algorithms can easily converge to inappropriate solutions because of improperly set search parameters, insufficient iterations, bad data, and just the nature of the specific problem. There is a strong need to involve a knowledgeable analyst in setting up and interpreting the results of this type of adaptive command and control and to examine the solutions continually for bad or inappropriate allocations of resources. An illustration of the need for this comes from some cases run with the SAGE algorithm during an analysis of NATO capabilities during the Cold War. We observed in a series of runs that Warsaw Pact aircraft were attacking empty NATO air bases. After some review of the cases, we learned that the reason was that we had required the algorithm to allocate all available aircraft to flying missions of one kind or another. Because all missions for the Warsaw Pact ground-attack aircraft involved heavy attrition of the aircraft with little payoff given the types of weapons we had allowed them to use, the algorithm found the "least costly" thing to do, which was to attack empty air bases. Once discovered, the fix to the problem was easily made. The bottom line is that models with these types of algorithms should not be used in a turnkey operation with little analyst interaction.

The Limitations of Automated Algorithms in Representing Human Behavior

Although we think this work with adaptive algorithms provides an important contribution to representing objective-based decisionmaking in simulation models, we must also remain humble and realize that there is much that we do not know about human decisionmaking under wartime situations and that we therefore have very limited capability to represent it in a simulation. Some of the aspects of human decisionmaking that are difficult to represent are

- learning from experience, which is gained from long-term training during conflict
- risk-taking and risk-adverse behavior and when such behavior should be represented

- using information beyond the simulation boundaries, such as knowledge of the particular opponents as individuals
- making decisions under uncertainty and filling in gaps in information with experience and intuition
- hedging decisions and when to hedge
- how humans use planning horizons
- decisionmaking in competitive and hostile environments.

Although additional research may shed some light on how to represent such behavior, it is also unlikely that we will be able to characterize many of these aspects in the near future. This is one reason it is useful to involve people in gaming situations as part of the analysis process.

Some Algorithmic and Implementation Issues for Representing Adaptive Planning in Models

There are specific issues associated with the algorithm we have presented. For one thing, there is no guarantee of convergence to a global optimum for either side. The search algorithms can get stuck in local optima, and it may simply take too long for the process to find an equilibrium solution. In practice, we have found that tuning the convergence parameters and the use of heuristics to guide the solution search provide practical approaches to assuring convergence to a good solution. However, there are no guarantees, and as we stated earlier, analyst involvement is critical to the quality of analysis from this type of approach.

There are other compromises in the approach to finding mixed strategies. It is possible for the solution choices to grow combinatorially from one planning period to the next. We have tried several approximate approaches to controlling this growth, including the use of mean estimates for the next state, selecting only the highest probability states, and computing a probabilistic state. In general, we have not encountered such state growth very often, but this may be because we have often analyzed forces with very asymmetric capabilities, which have only a few promising alternative strategies. More research needs to be done in this area, particularly as we attempt to introduce hedging and other actions in response to uncertainty.

7. Conclusions and Recommendations

In TLC, we implemented and investigated alternatives to four aspects of modeling we think are essential to improving theater level campaign analysis in the future: (1) how to create more-flexible structures to simulate the wide range of future scenarios and their associated uncertainties, (2) how to link to more-detailed models in an analytically valid way, (3) how to represent the maneuver of ground forces at the theater-campaign level, and (4) how to improve representation of adaptive behavior and aspects of command and control in this type of model. This research, while not resolving all of these issues definitively, has provided insights into some of the alternatives and has suggested some promising directions.

Observations and Conclusions

Flexible Structures

The choice of software structure is one of the most important decisions with respect to campaign-model development and is the foundation on which the simulation is built. The structure affects the level of resolution possible with the model, the efficiency in processing, the adaptability of the simulation to problems, the transparency of cause-and-effect relationships, and the ease with which future changes can be made. It also generally limits the analysis applications of the model.

Network Structures. The "generalized network" structure developed during our research with the TLC model eliminates the need for equal-size regions and regularity while maintaining the efficiency of the network structure. The generalized network is an extension of the regular network but reduces the coordination problems of multiple regular game boards with different-sized regions. As an extension of the regular network game boards, it still possesses much of the efficiency of those game boards but with added flexibility as well. The price for this added flexibility is the requirement for more analyst involvement in the network's development and the necessity of a GUI to create the networks.

Managing Time. One of the primary considerations of a simulation is the method used to manage time. In most theater-level campaign models, time is advanced in discrete steps by either the time-step or event-step method. The important advantages of the time-step method are its simplicity and the degree of control one can establish regarding the order of processing. The primary disadvantages of the time-stepped approach are the potentially large approximation errors in large time steps and possible inefficiencies of using very small time steps.

The efficiency advantages, the almost infinite ability to vary the time resolution, and the ability to avoid making implicit assumptions about processes occurring during fixed time intervals usually make event-stepped time management the method of choice. The fact that time steps can be represented as event steps in an event-stepped model (although slightly less efficiently) means that this approach provides a flexible underlying structure for simulating complex systems in which there may be natural time steps but that also have other asynchronous processes. TLC was implemented as an event-stepped model.

Random Processes. Many of the early and current campaign simulations are deterministic models. As the time between events or steps grows smaller and as more-detailed representations of individual objects become important and feasible, the deterministic approach encounters important limitations. Much about combat at higher resolution appears to be stochastic. At higher resolution, it is much harder to appeal to the law of large numbers, because the numbers are not large and the systems are not continuous. In general, nonlinear functions defined on expected values do not give the same results as the expected values of nonlinear functions of random variables.

The most common reason given for avoiding stochastic simulations is the number of independent runs required to achieve confidence in the results. But simulations of systems with high amounts of random variation tend to diverge significantly from their deterministic equivalents. In general, if the system under study has well-behaved random variation present in the quantities of interest, there is no reason to perform a deterministic simulation. The number of runs required of a stochastic or Monte Carlo simulation in this case may be estimated and will not be great. If the system under study is not well behaved, this is exactly the situation in which a stochastic simulation is required to perform analysis with reasonable confidence or at least to help bound the uncertainties. Choosing or building a deterministic model instead of a Monte Carlo model should depend on the nature of the system represented and not merely on expediency of processing. TLC was implemented as a Monte Carlo model.

Software Structures. The entities in a simulation can be modeled as objects in various object-oriented language structures or can be modeled with a top-down, hierarchical structure. We used two paradigms for event-stepped simulations: process interaction and event scheduling. In our experience, the process-interaction scheme initially appears to be simpler to model and more transparent to describe. This is because the entire process is described in one routine. The event-scheduling approach does not depict the whole process involving event processes. Instead, one must remember its components and abstractly visualize their interaction. On the other hand, the process-interaction scheme allows for much less control of the flow of the simulation than the event-scheduling scheme. This is because the programmer frequently has little direct control over the exact timing and order of execution of the "resuming" and "interrupting" functions using the process-interaction scheme. Instead, he must control them using "tie breaking" routines executed by the internal scheduler of the simulation. It is interesting to note that many simulation languages support both schemes and internally convert the process-interaction scheme to an event-scheduling equivalent.

Object-Oriented and Hierarchical Simulation. Conventional models generally become monolithic with a master procedure overseeing the execution of subprocedures performed by the entities of the simulation. Object-oriented modeling, as used in TLC, encourages a modular, decentralized approach to modeling.

The advantages of object-oriented modeling are maintainability of the existing model, extendibility of the model to other levels of resolution or other types of objects, reusability of existing objects, and evolvability of the model as one's understanding of the system increases. Most of the legacy campaign models have been built in the hierarchical style using non–object-oriented languages. Our experience with the development of TLC in an object-oriented language indicates that the modularity adds important advantages in terms of modifiability and replacement of objects, but it is no panacea. The hard work of modeling is in defining these data structures, processes, and process interactions to represent various military phenomena. These are complex regardless of the underlying modeling and software structure.

Distributed Modeling and Processing. Various approaches to distributed and parallel processing attempt to take advantage of multiple processors to speed up the individual runs of a simulation. For most analytic purposes, many trials are usually needed for sensitivity analysis or, in the case of Monte Carlo models, for statistical significance, so the problem is not so much getting a single run to execute quickly but to get a large number of runs completed. This leads to a distribution alternative known as *distributed runs*, involving the simultaneous execution of separate, independent runs of a simulation model on several computers. This approach takes advantage of local area networks and the explosion in the capabilities and numbers of scientific workstations available, without requiring specialized coding. The speedup achieved using distributed runs is equal to the number of computers used to make the runs, in that doubling the number of computers will halve the required time to complete the runs (except possibly for the initialization time costs). This was the approach we adopted for TLC.

Variable Resolution and Cross-Resolution Modeling

Variable-resolution modeling is the construction of a single model within which analysts can readily change the level of scope or detail at which the phenomena under study are treated. Cross-resolution modeling is the linking of existing models with different resolutions.

Variable-Resolution Modeling. In TLC, we developed structures that could be used to represent theater campaigns at various levels of resolution. The generalized networks, event-step processing, Monte Carlo approach, and object orientation could be used to provide high- or low-resolution representations of terrain, movement, advancement of time, and simulation objects. This does not mean that one could achieve a "zooming" capability with these structures. Once data and structures are initialized, the level of resolution is built in. That is, the structures permit one to select the level of resolution at the start of investigation of a particular scenario and problem and then to set up the model to run at that level of resolution.

It is possible to develop objects with different levels of resolution, but we found that too much of such selectable resolution can lead to a combinatorial problem in which all possible interactions must be modeled. All groupings of helicopters would need to know how to interact with company, brigade, and division objects, for example. The modeling of each of these interactions is nontrivial; and, to the extent that certain interactions are unlikely but possible, the selectable-resolution approach may require considerably more model development than would a model developed with just one level of resolution.

Cross-Resolution Modeling. A key aspect of campaign models is that much of the system and force effectiveness data must come from other, higher-resolution models. Values for surface-to-air, air-to-surface, air-to-air, and ground force–ground force effectiveness must usually be generated by more-detailed models run for a selected set of input cases. This linkage to higher-resolution models, or cross-resolution modeling, is essential for conducting high-quality analyses at the theater and operational levels. In general, most current approaches to aggregation, while appearing to be logical, have no good scientific or mathematical basis and cannot be expected to provide consistency across the different levels of aggregation, except within very loose bounds.

An important component of the TLC research involved developing good cross-resolution modeling, particularly for air-to-air, surface-to-air, air-to-surface, and ground-ground attrition. Some of the lessons learned in attempting to develop and use these attrition processes are as follows:

1. The strict dependency on other models increases the cost and difficulty of analysis with the theater-level model.
2. There is a need to perform more testing with the various approaches to model cross connection.
3. Real-time cross connection of models of different resolutions is probably not the solution for analysis, although there may be some utility for training.
4. Community sharing of data and testing of approaches would increase the practicality and quality of model cross connection.

Modeling Maneuver at the Campaign Level

During the Cold War, campaign models mostly neglected the maneuver aspects of warfare and performed dynamic balance assessments between opposing forces lined up in the pistons of the NATO defensive layer cake, organized around corps boundaries. An important goal of the TLC research was to model maneuver in a campaign model. Analysis of command, control, and intelligence capabilities, as well as the need to consider broad scenario uncertainties in which our forces might be initially outnumbered, means that a model capable of examining the current security environment should have the means to examine the important aspects of maneuver: the competitive, nonlinear movement of force and fire, the role of information, and the decision cycle. TLC's operational-level C^2 Planner was developed to simulate the execution of an operational maneuver plan by evaluating the likelihood of success of an operational plan, allocating reserve ground forces and interdiction assets to maximize the likelihood of achieving the highest-priority objectives, and planning for either the attacker, the defender, or both.

The C^2 Planner may be used in two-sided optimized simulations to represent opposing commanders' adaptive allocations and reallocations of resources in response to those of the opponent. In short, for each time period, each attack is evaluated in priority order to determine whether it is substantiated (i.e., whether there is a high probability of its success). If it is, the operational planner may widen the attack by adding another avenue to the attack and/or committing resources from echelons of lower-priority attacks (or operational reserves). If an attack plan cannot be substantiated, the planner may reduce its priority or adjust its phase lines. We found that this adaptive approach brings to the operational-planning process far more dynamic—and sometimes counterintuitive—but effective allocations of combat and noncombat resources.

Adaptive Resource Allocation

Models that do not react to information and the uncertainty about information by adapting strategies and resource allocations cannot show the value of C^4I assets and capabilities. Many current campaign models use nearly perfect intelligence for evaluating system allocation, and the decisions are inputs to the models. This makes it very difficult to evaluate new systems that attempt to improve information quality or timeliness. There is simply nothing to improve, and the hypothesized force-multiplier effect does not show up. Furthermore, if the decisionmaking does not adapt to new and better information, such as is the case in scripted or input allocations, no value will be shown for better C^4I. TLC has been designed to cope with the need for a model that can support the development of adaptive strategies for policy analyses.

There are other reasons such a capability is important for policy analysis. Adversaries are reactive and goal oriented. As a consequence, real strategies are highly interactive, and the outcomes of these interactions are often highly unpredictable—the consequences of actions may result in quite counterintuitive responses. An adaptive model can assist the analyst in recognizing critical uncertainties regarding threats and opportunities. Another reason is that policy analyses often need to consider how the "best" allocation of resources with respect to time (when, in what sequence), space (where) and function (CAS as opposed to BAI for fighter aircraft, reconnaissance as opposed to attack for helicopters, direct as opposed to indirect fires) will affect the value of those resources. Often, adaptations that an opponent can make will reduce the effectiveness of those resources, and often, a new capability should cause a change in one's own strategy. Finally, there may also be substantial manpower savings by not requiring human players to script decisions.

The SAGE algorithm is used to allocate air and other flexible combat assets, such as missiles and helicopters, optimally to meet theater objectives in TLC. SAGE consists of an algorithm within TLC's multiperiod conflict simulation that allows the user to specify whether overall objectives are to be minimized (e.g., own attrition) or maximized (e.g., attrition of opposing forces) and that engages in an automated search for the "best" strategies to meet the user-specified objectives. With the SAGE algorithm, the user specifies a measure of merit (e.g., targets or resources destroyed, final position of the forces, force ratio at the end of the conflict), and the methodology specifies a sequence of strategies for the allocation of resources of the two opposing sides.

SAGE solutions indicate the strategy that maximizes the marginal payoff of the attack aircraft, whether attacking land forces, air bases, or other targets. SAGE is used in TLC to allocate aircraft, long-range fires, and helicopters to mission types and interacts with the ground forces' operational planner described earlier. It is also used to provide situation-dependent target values for other algorithms.

TLC's combination of optimizing algorithms, artificial intelligence–based scripting rules, and decision tables contributes to treatment of operational strategy as an iterative and adaptive process, while enabling the user to retain control over the objectives and the extent of algorithmic optimization in the execution of the operational plan.

Recommendations

We encourage the campaign modeling community to utilize the lessons learned with this model about approaches to flexibility, model structure, cross-resolution modeling, maneuver representation, and adaptive decision modeling. We believe the research has defined, implemented, and tested important approaches to achieving scenario flexibility and variable resolution in time, space, objects, and processes. Much of this flexibility can be achieved while retaining the efficiencies of less flexible structures by using generalized networks, event processing, and object-oriented modeling.

We are only beginning to understand the issues and limitations of the cross connection of models and the approaches to aggregation and disaggregation in achieving consistent results. We know that consistent aggregation can often be done, but a consistent approach requires research with a high-resolution model and is often model and data dependent. We also know that most current approaches are ad hoc and have not been sufficiently tested. *We suggest that organizations publish their approaches to model cross connection along with testing that has been done, to improve the knowledge base in this area.*

The input effectiveness data for the algorithms of a campaign model generally require a considerable number of cases from high-resolution models to span the space of situations encountered in the campaign model. This is a tedious and time-consuming process, somewhat beyond the resource limitations of most campaign analysis organizations, at least to do it right. Our conclusion is that successful cross-resolution modeling, in which detailed results of high-resolution models feed campaign models, will require data sharing across organizations to populate the space of cases required. We recommend that data development be organized within the Department of Defense. *We note that this involves the identification of models to be used, cases to be run, data to be stored, and aggregate approaches to be defined; contextual description of the data; and development of a process to make the results available to analysis organizations.*

Our research addressed new ways to represent combat phenomena that we think we understand. However, there are many things we do not understand. These cannot be resolved merely by declaring that a new model will "include" those phenomena. *We believe specific military-science research should address how to model poorly understood phenomena, such as information warfare. This research would have a goal of creating consistent models of the phenomena that cross levels of resolution from high to low.*

It is our belief that simulating adaptive resource allocation is a key element to understanding the importance of information and command and control systems. The scripted decisions inherent in many of the legacy campaign models simply do not react to an information war. We developed and tested two approaches to adaptive resource allocation, and those are described in this report. We expect that ADS experiments with people in the loop could further understanding of decisionmaking under uncertainty.

Finally, we note that there are many purposes for modeling and simulation. In some cases, it is to help structure and think through problems. In other cases, it is to perform less-expensive testing or training. The use of models for analysis, our primary focus, implies that the models should be transparent, permit sensitivity analysis, be repeatable, and be usable within the time and resource constraints of the analysis requirement. On the other hand, modeling for other purposes may impose requirements for realism, rapid changeability, interactivity, etc. It is unlikely that one model will serve all purposes, let alone be useful for the full range of analytic activities. In our discussion of multiple-resolution features, we show that the interaction combinatorics between objects get in the way of having lots of objects at various levels of resolution interacting with all other objects. In fact, inordinate amounts of development time may be required to deal with process interactions that would never occur or are unlikely to occur in such a general-purpose model. *We recommend that any campaign model be developed with a limited vision of use and range of applications and that multiple* *models be developed to broaden the range. One model designed for too many purposes is not likely to suit any of them well.*

An important thread in this report is that many of the model innovations we suggest add a greater burden or reliance on the analyst. To use a model like TLC, with its flexible structures; variable-resolution features; and adaptive, objective-based resource allocation, the analyst cannot simply turn the crank. The analyst is forced to be intimately involved in the data and structure—a good thing, we think.

Appendix

A. The Calibrated Differential Equation Method for Estimating Ground-Combat Attrition

Background

Modeling close-combat attrition in a highly aggregated model while preserving flexibility of representation and transparency to the user has long challenged analysts. This particular problem has a number of aspects. Specifically, analysts must be sensitive to the actual resources employed during a battle; in particular, they must consider any synergies that might exist between those battle resources and supporting systems. Also, analysts need to ensure that the model is sensitive to the battle situation (e.g., environment, battle type, doctrine) and to munitions losses and use. In addition, the model should be heterogeneous, allowing the analyst to differentiate between resources in battle. For example, an analyst will frequently want to determine the effect of trading off cheap, plentiful resources against scarce, expensive ones. Analysts may also want the model to be able to integrate tactical air and indirect fires. Finally, the methodology should be based on observable data drawn from historical battles or exercises or from more-detailed simulations, rather than being made up completely from hypothetical data. The model should be able to extend beyond the available data when those data are inadequate. TLC addresses this problem with a weapon-on-weapon combat adjudication process called Calibrated Differential Equation Methodology (CADEM). CADEM is a RAND-developed extension of the Attrition Calibration (ATCAL) methodology developed by the U.S. Army Concepts Analysis Agency (Louer, 1991).

Objective

This appendix explains the design and operation of CADEM, which addresses many of the concerns raised above. Specifically, CADEM uses attrition data generated from higher-resolution models or, if available, historical or exercise data. Second, it allows use of judgment to adjust for a lack of data or the inadequacy of the available data and to incorporate hypothetical systems into the model. Third, it directly considers interactive effects, such as weapon system dependencies and shortages of munitions, and allows the representation of fratricide and suppression. Finally, it uses, wherever possible, a graphical interface to make the procedure more transparent to the user.

Design Goals for CADEM

Desired Characteristics of the Attrition Process

Concerns with the attrition process used in current models led us to attempt to design a model more sensitive to the resources employed in the battle and to any synergies that might exist between those resources and supporting systems. In addition, we wanted this attrition process to be sensitive to the environment, type of battle, various doctrines employed, and so forth. We wanted to integrate tactical air and indirect fire, if possible. Further, since we often trade off quantity against quality, we wanted the attrition methodology to be heterogeneous and to differentiate among basic resources during battle. We felt that our methodology should rest upon observable exercise or historical data or upon data obtained from high-resolution simulations, rather than being made up completely from hypothetical information, but we also wanted to maintain the ability to extend beyond these available data.

Description of the Attrition Process

Before proceeding with a discussion of the CADEM process, it is useful to look at the attrition process itself. To do so, we need to define two terms: resource and situation. Low-resolution models normally use battalion-sized units and larger. Those units maintain lists of resources. A resource can be any quantifiable item, such as numbers of weapon systems, score values of weapon systems, days of supply, and so forth. Because this is an attrition methodology, if a resource is included in the model, it either ought to be subject to attrition or to cause attrition, directly or indirectly.

A second definition pertains to what we call a situation, which is essentially everything else of significance, such as the weather, terrain, battle posture, or a given doctrine. We view what we call a situation as constant, because we assume that the situation either changes so slowly that it might as well be constant or that it changes so dramatically, such as going from day to night, that it becomes a new situation. We treat the dramatic change as a constant up to the point of change, then switch to another regime.

We model attrition as a function of resources and situation. Figure A.1 depicts three-dimensional axes, composed of resources (X), a situation (S), and attrition (dX). When plotted as a triad (X,S,dX), the values of the function form a response surface. For each intersection of various numbers of resources and the situation, we want to capture the attrition at time t over an infinitesimal amount of time around t. Attrition data should fall on this surface.

But the process may not be so straightforward, because the resulting data might be stochastic or the result of a stochastic process. We thus have a stochastic response surface when each dX is drawn from a distribution that depends upon the resources, X, and on the situation, S. Figure A.2 suggests the stochastic nature of this response by depicting attrition as a distribution curve sitting on the response surface.

The process may become even more complicated. The question the analyst faces is whether, given several observations of resources, situations, and corresponding attrition confounded with random error, he can calculate attrition for an unknown or for a nonpredictable number of resources at a point in time. Figure A.3 poses this question by depicting attrition observations from two sets of data at (X_1,S_1) and (X_2,S_2) and extending it to the attrition, ?, for a new situation, S, and set of resources, X.

The CADEM Approach

The CADEM process represents our answer to the question. We drew many ideas from a methodology called ATCAL used by U.S Army Concepts Analysis

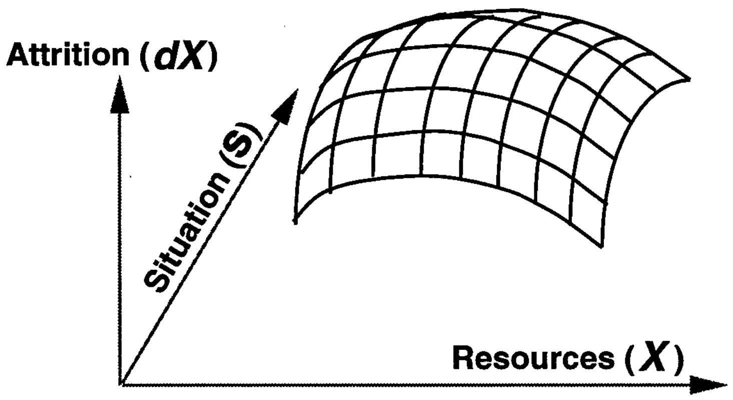


Figure A.1—Attrition Is a Function of Resources and Situation

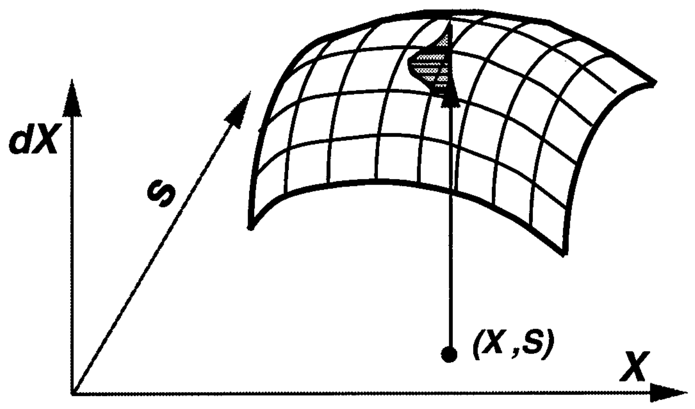


Figure A.2—Attrition May Be Stochastic

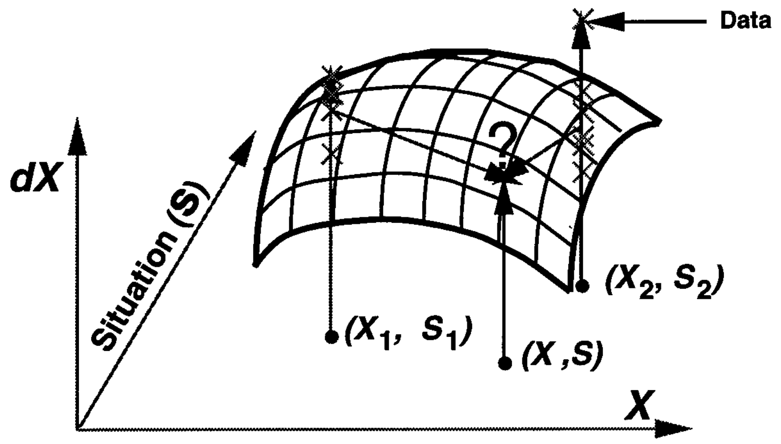


Figure A.3—How Best to Use Available Attrition Data

Agency.[1] The CADEM approach uses attrition data generated from high-resolution models or, if available, historical or exercise data. Then, the application of judgment can adjust for lack of, inconsistency in, or inadequacy of the data and can incorporate hypothetical or notional systems.

Also, the CADEM approach may directly consider interactive effects between weapons and between munitions and may even allow representation of fratricide and suppression. Whenever possible, we try to use a graphical interface based on a spreadsheet implementation of the methodology to make this procedure more transparent to the user.

[1]See Analysis Support Directorate (1983) for more information.

Figure A.4 depicts the overall process, which consists of several primary steps. In brief, these are

- drawing data from killer-victim scoreboards and running a series of calculations to compute calibration parameter sets, one for each situation that the model will likely encounter (e.g., different terrain)
- extending the calibration parameter sets to situations inadequately covered by the killer-victim scoreboards
- selecting a parameter set for the specific situation
- forming an attrition matrix of the coefficients of a linear system of differential equations
- adjusting the attrition matrix for the mix of resources in the battle (e.g., fewer tanks, more infantry)
- augmenting the system of differential equations to represent relationships between resources not captured in the killer-victim data
- calculating the attrition by numerically solving the system of differential equations.

Each of these steps will be described in the following subsections.

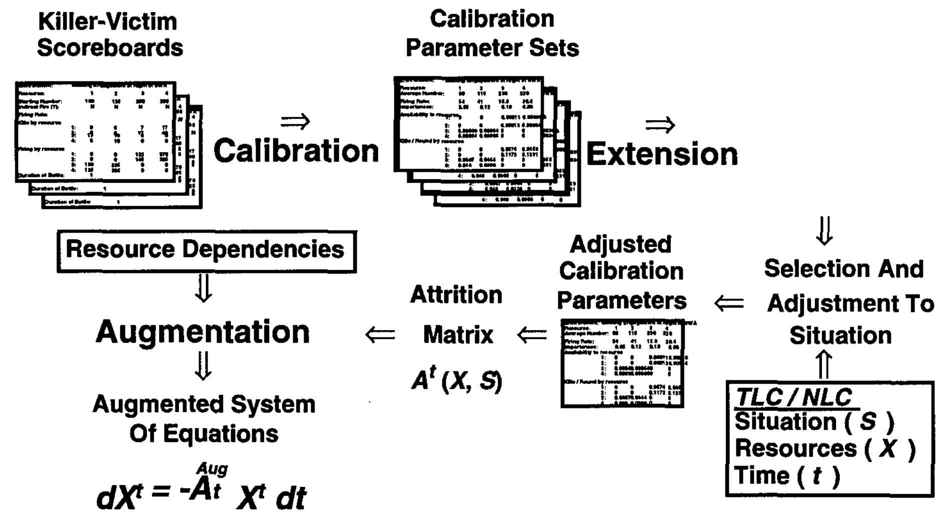

Killer-Victim Scoreboards
Calibration
Calibration Parameter Sets
Extension
Selection And Adjustment To Situation
TLC / NLC
Situation (S)
Resources (X)
Time (t)
Adjusted Calibration Parameters
Attrition Matrix
$A^t(X, S)$
Resource Dependencies
Augmentation
Augmented System Of Equations
$dX^t = -\overset{Aug}{A^t} X^t dt$

Figure A.4—The Complete CADEM Process

Calibration

Figure A.5 depicts the CADEM calibration step and its relationship to the entire process.

Given a set of killer-victim scoreboards, which could come from high-resolution models, historical battles, National Training Center data, etc., the process goes through a calibration step, which produces calibration parameter sets.

Table A.1 presents an example of a killer-victim scoreboard. Each scoreboard starts with a description of the situation under which the data were obtained. Each scoreboard contains the duration of the battle in units of time consistent with the representation of time in the model. It lists the initial number of each type of resource. It then describes what killed each type of resource. In this case, resources 1 and 2 shot at 3 and 4 and vice versa. For example, resource type 2 killed 17 of resource type 3 and resource type 3 (in its role as shooter) killed 10 of resource type 2 (in its role as target). Additional data, such as whether each resource shoots with direct or indirect fire and the firing rate of the resources, may also be available. If such data are not available, they may be approximated. Simply knowing who killed whom does not provide enough information. We also need to know who shot how much at whom, and thus we have a requirement for the last part of the killer-victim scoreboard, a firing table of who fired how many times at whom.

These data allow computation of the basic components of attrition for this methodology. The basic components to be calibrated fall into two parts: kill

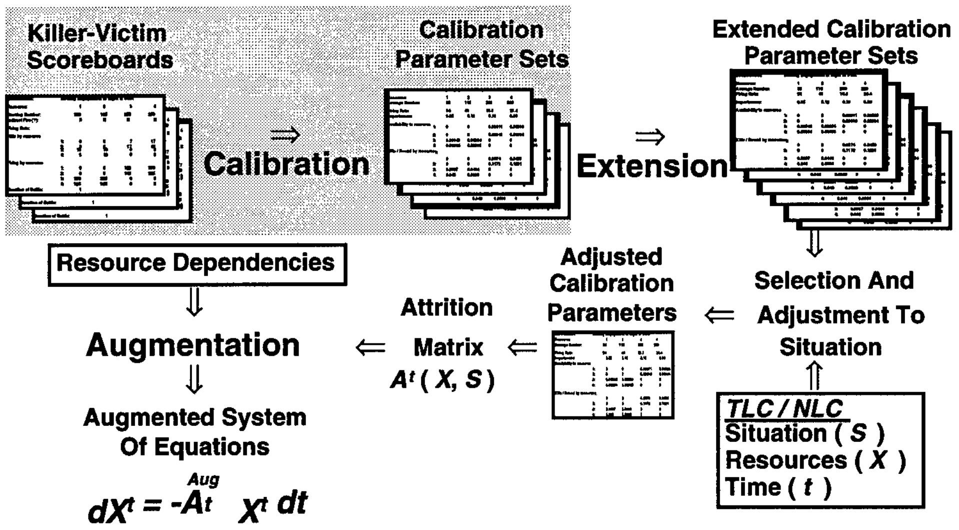

Killer-Victim Scoreboards
⇒ Calibration
Calibration Parameter Sets
⇒ Extension
Extended Calibration Parameter Sets
Resource Dependencies
Augmentation
Augmented System Of Equations
$dX^t = -A_t^{Aug} X^t dt$
⇐ Attrition Matrix $A^t(X, S)$ ⇐
Adjusted Calibration Parameters
⇐ Selection And Adjustment To Situation
TLC / NLC
Situation (S)
Resources (X)
Time (t)

Figure A.5—Calibration Step

Table A.1

Killer-Victim Scoreboard

Type of Battle:	**Meeting Engagement**			
Duration of Battle:		1		
For Resource:	1	2	3	4
Initial Number:	100	125	250	250
Kills by 1:			7	17
Kills by 2:			17	40
Kills by 3:	13	10		
Kills by 4:	6	10		
Type of Fire:	Direct	Direct	Direct	Direct
Firing Rate:	—	—	—	—
Fires by 1:			122	370
Fires by 2:			145	325
Fires by 3:	150	225		
Fires by 4:	125	325		

rates and allocations of fire. Each of these has, in turn, two subcomponents. Kill rate is composed of kills per round and the firing rate. Kills per round is a fairly easy calculation. It is the number of rounds fired divided into the number of kills. The firing rate is not as easy to calculate, and it is explained below. Allocation depends upon two subcomponents, the availability of targets and firing priority or doctrine. The result is a set of values logically related to weapon capability, mobility, line of sight, force distribution, doctrine, etc. From these values, logical arguments can be made about how these things change as the doctrine changes. Because these data are more objective, it is frequently easier to reason about these entries than those in the killer-victim scoreboard or such things as, say, scores.

Kill Rates

As mentioned, the kill rate consists of two parts: kills per round and the firing rate. Table A.2 depicts the kills per round for the data in Table A.1.

Firing Rates

The firing rate is the maximum rate at which a weapon can fire with no consideration given to target availability. Stated another way, it is the rate of fire

Table A.2

Kills per Round

For Resource:	1	2	3	4
Kills Per Round by 1:			0.0574	0.0450
Kills Per Round by 2:			0.1172	0.1231
Kills Per Round by 3:	0.0867	0.0444		
Kills Per Round by 4:	0.0480	0.0308		

possible with an infinite number of targets available. The rate does include such operational constraints as resupply time, rest time, and operating procedures. These can be specified or calculated from the killer-victim scoreboard. The firing rate is not the test-bed or maximum rate of fire possible by the weapon. It is the operationally possible rate, unconstrained by targets.

Allocation of Fires

The second component of the attrition calculation, allocation of fire, is the likelihood of a given shooter firing at a specific target. It is the probability that at least one target is available to be shot at multiplied by the likelihood that no other, higher-priority target is present. Availability determines what resources may be fired upon, and the priority scheme determines firing decisions when multiple target types may be available.

The key element in determining allocation of fires, as Figure A.6 suggests, is target availability. Target availability is the probability of engagement by one shooter and one target on a given firing cycle. With the high number of shooters, targets, and firing cycles, these values tend to be very small.

In determining allocation of fires, we also consider three types of firing priorities. These can reflect doctrine and other considerations. The first type is "first seen, first shot" which allocates fire proportional to target density. This type of priority means we cannot really distinguish at whom we fire. A second type of priority is that specified by doctrine or the analyst. For example, doctrine might hold that command vehicles receive first firing priority. A third priority scheme rests on computed target worth or the importance of a target to be engaged. The importance of a target in our model relates to its ability to kill as a shooter. If a target can kill a large number of valuable shooters, it has a high target worth. The priority of a target to a shooter then equals the target's importance multiplied by expected kills per round if the shooter were to fire at it. These

The likelihood that

- **one *specific* target may be fired at by**
- **one *specific* shooter during**
- **one *specific* firing cycle**

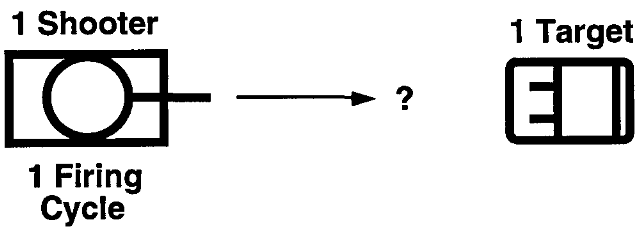


Figure A.6—Target Availability

priorities can be computed from the data or specified in the model, designating that a given target type will be engaged first, another second, and so forth.

If r denotes the number of resources, I_s denotes the importance of one type s resource, Δt denotes the length of the time interval, N_s denotes the number of type s resources, $F_{s,j}$ equals –1 if s and j are on the same side and 1 otherwise, ΔN_j denotes the number of type j resources killed and $\Delta N_{s,j}$ denotes the number of type j resources killed by type s resources, the importance of resources may be computed by the system of equations:

$$I_s^3 = \sum_{j=1}^{r}\left[\left({}^{\Delta N_{s,j}}\!\Big/_{N_s \Delta t}\right)^3 \left({}^{F_{s,j} I_j \Delta t}\!\Big/_{N_j}\, \Delta N_j\right)\right], s = 1 \ldots r \qquad \text{(A.1)}$$

Each equation relates the importance of a resource to its contribution to the attrition of other resources.

Table A.3 presents an example of various calibration parameters. The combination of the information in Tables A.2 and A.3 is a calibration parameter set. We have, for each resource, the average number alive during the battle assuming an exponential drawdown; the firing rate (in this case, calculated from the data); the importances; and the availabilities. This table shows that these availabilities turn out to be extremely small. Of course, when multiplied by the

Table A.3

Calibration Parameters

Type of Battle:	Meeting Engagement			
For Resource:	1	2	3	4
Average Number:	90	115	238	220
Importances:	0.05	0.12	0.10	0.08
Firing Rate:	54	41	15.8	20.4
Availability to 1:			.00011	.00036
Availability to 2:			.00013	.00034
Availability to 3:	.00048	.00054		
Availability to 4:	.00034	.00065		

number of shooters, the number of targets, and the number of firing opportunities, the resulting product can be quite large.

The calibration parameter sets represent the output of the calibration step. The next subsection goes into more detail as to how the calibration parameter sets can be extended when the data are inadequate.

Extension and Selection

We now turn in more detail to the steps of the model that deal with extending the calibration parameter sets to situations for which data may be inadequate or unavailable and selecting the appropriate calibration parameter set to use for a specific situation. As part of the selection step, the technique for the formation of the attrition matrix from the selected calibration parameter set will be presented. Figure A.7 displays the CADEM process and highlights these steps.

Extension

A collection of calibration parameter sets may be extended using due consideration and logic, experience, judgment, or just "what- ifs." For instance, the analyst may have a calibration parameter set that includes an M1A1 tank as a resource. But he wishes to include M1A2 tanks, which might, for example, feature an automatic loader and extended engagement range. That set is developed in the extension step by modifying the M1A1 parameters in the original set to produce the M1A2 version.

Figure A.8 illustrates the extension step. The calibration parameter sets can be extended to include other data based on such things as experience, history, logic,

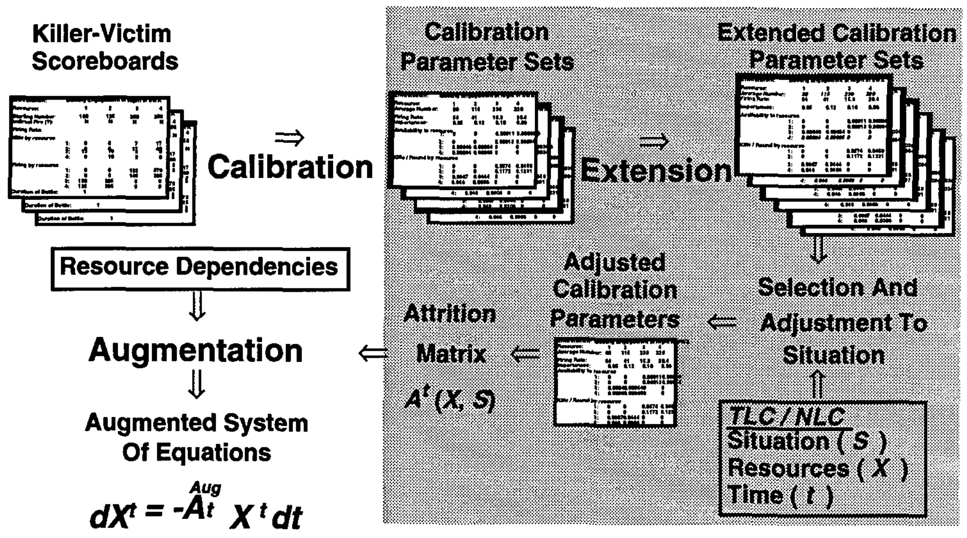

Killer-Victim Scoreboards
⇒ Calibration
Calibration Parameter Sets
⇒ Extension
Extended Calibration Parameter Sets
Resource Dependencies
Augmentation
Augmented System Of Equations
$dX^t = -\overset{Aug}{A_t} X^t dt$
Attrition Matrix
$A^t(X, S)$
Adjusted Calibration Parameters
Selection And Adjustment To Situation
TLC/NLC
Situation (S)
Resources (X)
Time (t)

Figure A.7—Extension and Selection

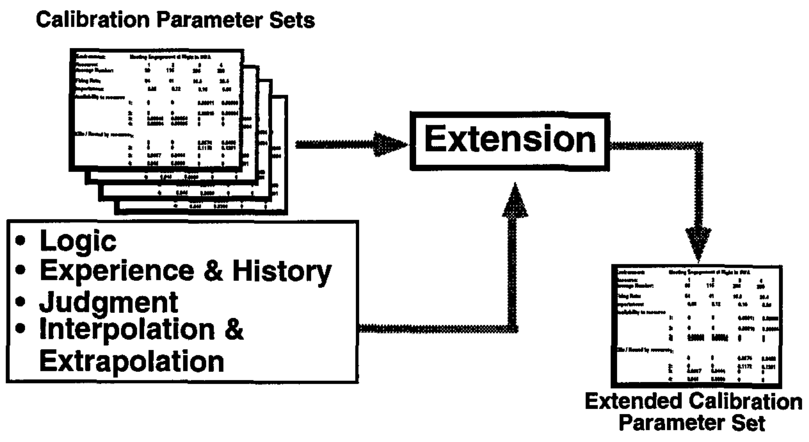

Calibration Parameter Sets
Extension
• Logic
• Experience & History
• Judgment
• Interpolation & Extrapolation
Extended Calibration Parameter Set

Figure A.8—Extension of the Calibration Parameter Sets

judgment, and so forth. For example, suppose a decision is made to install a faster loader into a notional or hypothetical tank. Empirical data to serve as a basis for enlarging the killer-victim scoreboard database would, of course, not be available. But the calibration parameter sets may be modified based simply on logic. In this case, the firing rate of an extant tank, say the M1A1, could be increased by 20 percent, and all other characteristics of the weapon would remain unchanged. While this is obviously not a perfect solution, it would generate some data for analysis.

By way of a second illustration, an analyst might wish to investigate battles in a theater other than the one from which the killer-victim scoreboards were drawn. Assume that this analyst changes the venue from central Europe to Kuwait. Some differences are obvious. Terrain is far more open in Kuwait, and greater visibility will normally result in engagements at longer ranges. The practical effect is that additional targets will be available to the weapon systems that can exploit these longer engagement ranges. Given these deductions, the analyst would increase the availability parameters for the targets of the weapon systems that could take advantage of the longer lines of sight. But longer engagement ranges normally mean some decrease in accuracy, so the analyst would want to decrease the parameter for kills per round for the long engagements.

These first two examples illustrate CADEM's capability to extend calibration parameter sets to situations for which adequate empirical data may not be available. Ordnance experts could provide information about automatic loaders, and history contains a great deal of information about desert tank battles. But some data are very difficult to find. Suppression is a case in point. Exercise data contain little information about the effects of suppression, but most agree that it has a definite effect on the outcome of the battle. CADEM parameter sets could be extended to account for suppression effects as follows.

Suppression occurs through both offensive and defensive measures. Offensive measures would include such things as suppressive artillery fire, which has the effect of lowering the enemy rates of fire. Use of smoke to hide friendly movement, an example of defensive suppression, has the effect of reducing the availability of targets to the enemy. And the enemy kills per round should decrease, because even when seen, the targets tend to be more fleeting. Thus, the addition of suppression results in changes to the original calibration parameter sets: Enemy rates of fire, kills of friendly targets per enemy round, and availabilities of targets to the enemy should decline. In addition, the effective rate of fire of friendly artillery should be decreased to compensate for the delivery of suppressive fire. Of course, the analyst should always seek better data from killer-victim scoreboards, just to ensure the logical assumptions are accurate.

Selection

While extension modifies calibration parameter sets exterior to the TLC model, selection focuses on modifications during the execution of the model. Extension looks outside the model to such things as history and judgment. Selection looks inside the model and uses mathematical algorithms. During the execution of the model, each specific battle requires that a choice based upon the battle situation be made from available calibration parameter sets. The process can be carried out by a combination of decision rules and stochastic sampling. For example, an analyst may have 25 Desert Storm calibration parameter sets, of which eight may apply to a specific situation occurring in the model. As described by decision rules, the model could choose the one of the eight deemed to be the most likely or could average the applicable sets. Or the model may "flip a coin" for each battle and pick one of the sets at random, each with probability 12.5 percent. The latter procedure captures some of the stochastic variability inherent in the original data. Figure A.9 outlines this step.

As another example of stochastic sampling, assume the analyst has a large number of calibration sets run for the same situation, but they vary considerably because different analysts controlled the runs. One approach would be to discard the more extreme cases and average the rest. However, the problem with an average set is that it often has little chance of ever occurring. To illustrate, in 100 battle runs, Blue might win 80 percent and lose 20 percent. But on average, Blue would win every time. A better approach would be to keep all the boards and select from them randomly.

Calculating the Attrition Matrix

After the selection step (which is performed for each new battle), the analyst has one set of calibration parameters. These calibration parameters are used in the

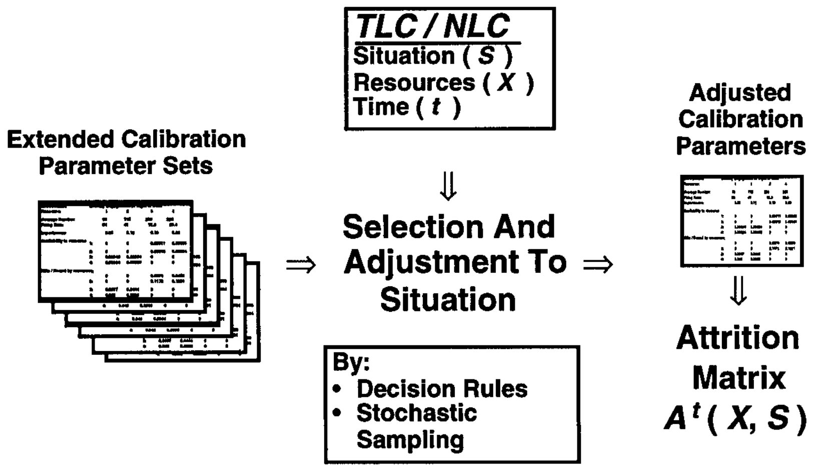


Figure A.9—Selection and Adjustment

model to produce an attrition matrix, which is a function, as mentioned, of the number-of-resources vector, X, the situation, S, and the time, t. The attrition matrix is adjusted for the mix of resources in the battle. The procedure uses a form for attrition very much related to a Lanchester model of attrition (Taylor, 1983). This form is a linear system of differential equations with time-varying coefficients:

$$\left.\frac{dX}{dt}\right|_t = -A_t X_t \qquad \text{(A.2)}$$

The A_t matrix in the above equation is known as the attrition matrix and may vary over time. The advantages of such an approach are that the analyst can represent most attrition processes and that the procedure is well known. The disadvantages are that, even though the representation of attrition is good, it may be difficult to solve numerically and that it is subject to abuse. If the attrition matrix does not vary with time or resources, the Lanchester Square Law for attrition holds. Figure A.10 presents a stylized version of the technique used to determine the attrition matrix.

The figure shows the surviving resources plotted on the vertical axis (Resources) versus the horizontal axis (Time), as if the resource vector were one dimensional. For a initial number of resources, X, and a fixed time period of specified length, Δt, the technique iteratively attempts to find a point, X', at which the derivative of the curve equals the slope of the curve. If it does find such a solution, the elements of the attrition matrix are the negatives of the coefficients of the resources for the derivative. The components of the point at which the derivative

$$\Delta X^t = -A^t X' \Delta t$$

$$X' = -\Delta X^t / \mathrm{LN}(1 - \Delta X^t / X)$$

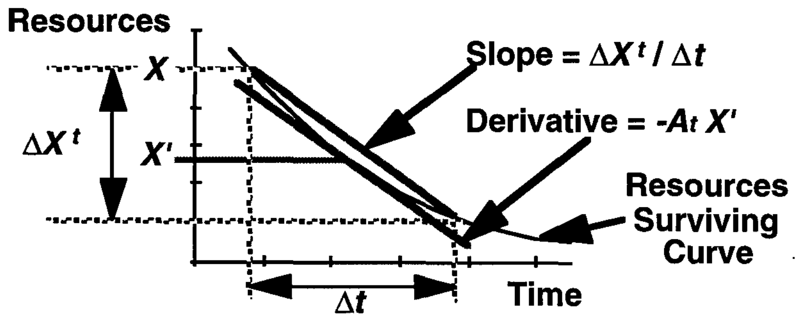


Figure A.10—Attrition Matrix Computation

is computed are the average number of each resource over the time interval if an exponential drawdown is assumed. At each iteration, the slope from the previous iteration determines a new average, which in turn allows a new derivative to be computed, which yields a new slope. The process iterates until numeric convergence occurs. Perfect convergence may not occur, but the difference should be small. If the time interval is short, convergence will occur after a few iterations.

As an example, we will compute the attrition matrix on the interval from 0.0 to 0.5. While this interval is much longer than that used in practice, it lends itself to computational ease. We will also assume that there are initially 50 resources of type 1, 125 of type 2, 250 of type 3, and 500 of type 4. Iterating through the procedure yields the number killed, the average number alive during that time interval, and the attrition matrix presented in Table A.4.

Examples of the Extension and Selection Steps

In this subsection, we present executions of the CADEM process for a time interval of length 1 using the killer-victim scoreboard and calibration parameter set developed in the last subsection, as well as extensions to that calibration parameter set. Technically, if the original data are entered, the results should mirror the original killer-victim scoreboard. In fact, that is exactly what occurs in CADEM. It is an attractive feature of the model that, if the calibrated data are

Table A.4

Example Solution

Type of Battle:	**Meeting Engagement**			
For Resource:	**1**	**2**	**3**	**4**
Initial Number:	50	125	250	500
Average Number:	46.5	117	245	473
Killed:	6.82	16.17	9.18	52.51
Killed by 1	0	0	1.57	9.23
Killed by 2	0	0	7.61	43.28
Killed by 3	3.48	5.25	0	0
Killed by 4	3.34	10.92	0	0
Attrition Matrix:	0	0	0.028	0.014
	0	0	0.043	0.046
	0.067	0.130	0	0
	0.397	0.742	0	0

entered, a calibrated solution emerges. We will extend the calibration parameter set for two cases: reducing the firing rate of a resource and introducing fratricide.

Figure A.11 shows the effect on attrition, after one time unit of battle, of reducing the rate of fire for a given weapon system. In this situation, resources of type 1 and 2 fire at resources of type 3 and 4, and vice versa. The left side of the figure shows the base case, which uses the calibration parameter set derived in the previous section. The right side of the figure shows the results of reducing the firing rate of resources of type 3 by one-half. In the base case, resources of type 3 accounted for most of the kills on resources of types 1 and 2. With a reduced rate of fire, as expected, fewer resources of types 1 and 2 are killed, which might be termed the main effect. But the model also reveals the interactive second-order effects. The second-order result stems from the fact that we allow weapon systems to evaluate the worth of targets. Because the resources of type 3 are firing only half as fast, the importance of those resources declines enough so that resources of types 1 and 2 reallocate their fire toward resources of type 4. These effects are presented in Figure A.12.

This figure shows that, when resources of types 1 and 2 reallocate their fire, fewer resources of type 3 are killed. This effect, of course, is fairly intuitive.

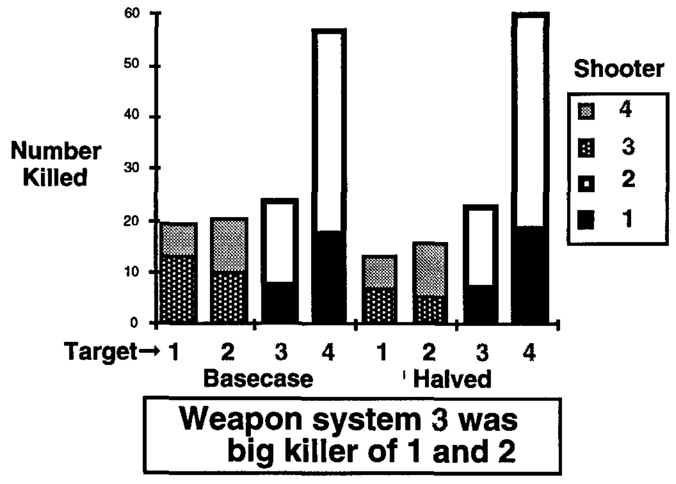


Figure A.11—The Effect of Having the Rate of Fire of Type 3 Resources

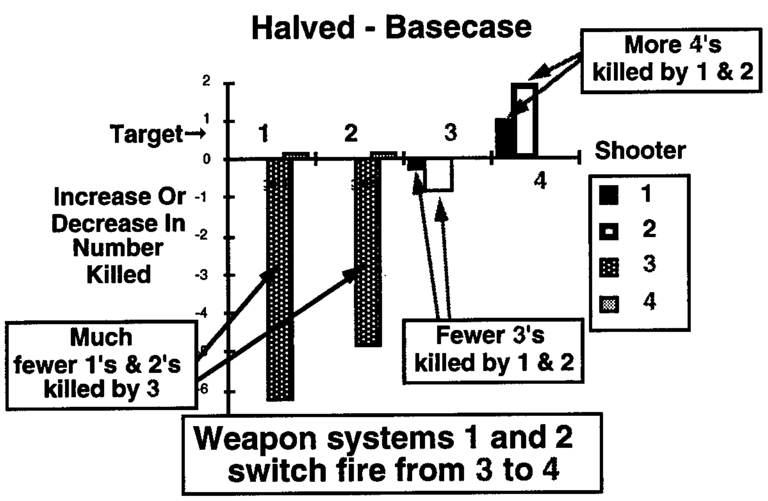


Figure A.12—Changes Due to Reduced Fire Rate of Type 3 Resources

Weapons that do not shoot tend not to draw fire. The point is to highlight that the model captures this effect (and many do not).

Unlike traditional models that reflect only Red-on-Blue killing, CADEM allows fratricide, Red-on-Red or Blue-on-Blue killing. Figure A.13 displays an incidence of fratricide. The third bar from the right reflects a kill of resources of type 2 attributable to resources of type 1. The original calibration parameter set was extended to a set of calibration parameters that had nonzero availability and kills-per-round entries for resources of type 2 to resources of type 1.

Numeric Integration

The entire CADEM process has been encoded in a number of ways. It has been programmed in an object-oriented format using MODSIM II and the C languages. A common numerical integration algorithm is used to solve the system of differential equations. Additionally, the process has been encoded into a spreadsheet model with extensive graphics. The latter feature allows the analyst to change parameters and see the results quickly. Presently, only two versions of the spreadsheet model exist, one for the two-resource on two-resource situation and one for the seven-on-seven situation. The latter version

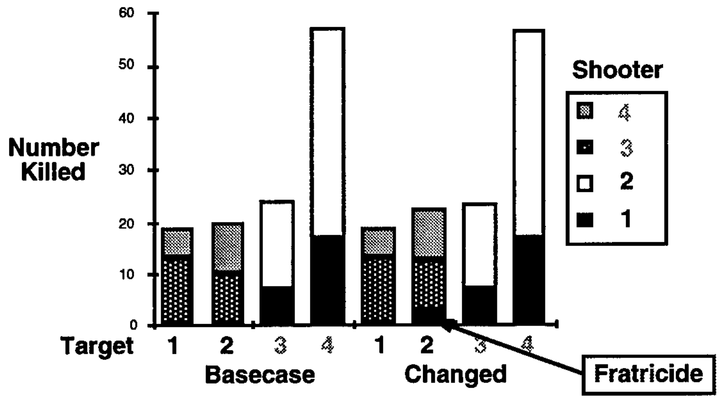


Figure A.13—CADEM Models Fratricide

runs rather slowly, but both allow the analyst to get an initial sense of the implications of an extension to a parameter in CADEM.

Both encodings of the process numerically solve systems of differential equations. In solving this type of system, the key is to be able to calculate the derivatives at any of the necessary points in time. These derivatives are approximated in CADEM by the procedure described in the previous section as differences over short time intervals. Short time intervals are better than long time intervals, because the shorter the interval, the faster the procedure converges and the more accurate it becomes. The ATCAL procedure mentioned earlier solves the differential equations over a fixed time step of unit length (however that length is defined by the modelers) using a modified Euler approach. CADEM uses short time intervals and a Fourth Order Runge-Kutta solution technique with adaptive step-size control of error. If higher efficiency and/or higher accuracy are major considerations, other numerical integration techniques could be used. Although CADEM computes the derivative over short time steps, it computes attrition curves over a long time interval. The solution techniques used in CADEM are most efficient when doing so, because then they may determine the variability over time in the solution to the system of equations. CADEM uses a feature called time-warp to assist it in that process. Time-warp allows us to compute into the future, but if something dramatic happens that invalidates part of the data as the rest of the model is catching up, we roll back CADEM. A rollback is performed simply by halting the CADEM solution at a point, correcting the data, and solving from that point on using the corrected information. We thus let the numerical solutions of the differential equations run efficiently into the future and roll back as necessary.

TLC is a discrete-event model, which means that it works most efficiently when predictions may be made about when certain events are going to occur and then those events are scheduled to happen. This procedure is more efficient than inquiring at every time step or so whether an event has happened. If attrition has been predicted, it is then possible to anticipate various events—when the force ratio falls below 3:1, for example—and then schedule those events to occur. If a rollback does not occur, it is not necessary ever to check or schedule the event again. When an event (the force ratio reaching 3:1) occurs, the model will call into being whatever performance has been specified for the event. The fact that a solution exists in the future allows the resource monitor to predict when significant events are going to happen.

Figure A.14 displays the change in resources over time. Note that the space between data points on the resources curves in Figure A.14 gradually lengthens as time increases and that the curves become more linear in time, indicating that longer time intervals are being used for the solution technique to maintain a constant error bound.

Augmentation

The last step of the CADEM process, as indicated in Figure A.15, involves the augmentation of the differential equations for dependencies. Augmentation simply means adding extra equations to change or supplement the way things were represented in the original data to match current purposes. For instance, dependencies of attrition upon some resources—ammunition consumption, multiple systems per vehicle, exogenous attrition—all these types of things can be incorporated into the system of equations, arriving at an augmented attrition matrix. This augmented attrition matrix contains the coefficients in an augmented system of differential equations, which are solved to give surviving resources.

Augmentation represents one of the more attractive features of CADEM. An analyst may have other effects that do not come from the killer-victim scoreboards but still wants to represent them in the attrition process. If we know they exist and can be represented by differential equations, they can be entered into CADEM, because it is not a closed system. ATCAL, for example, solves a

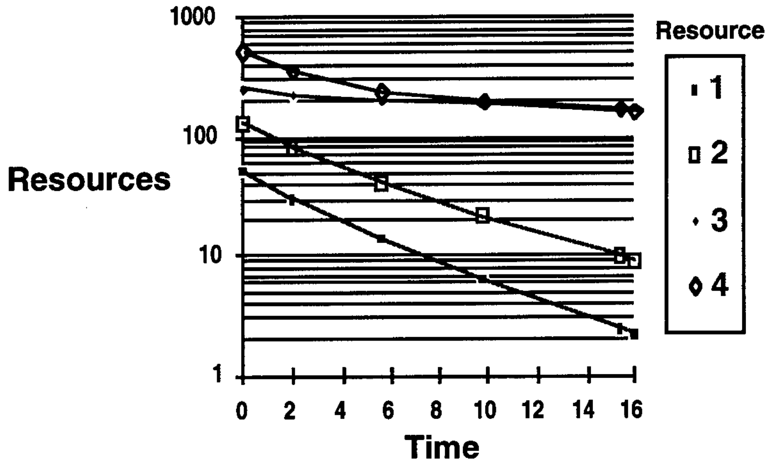


Figure A.14—Resources over Time

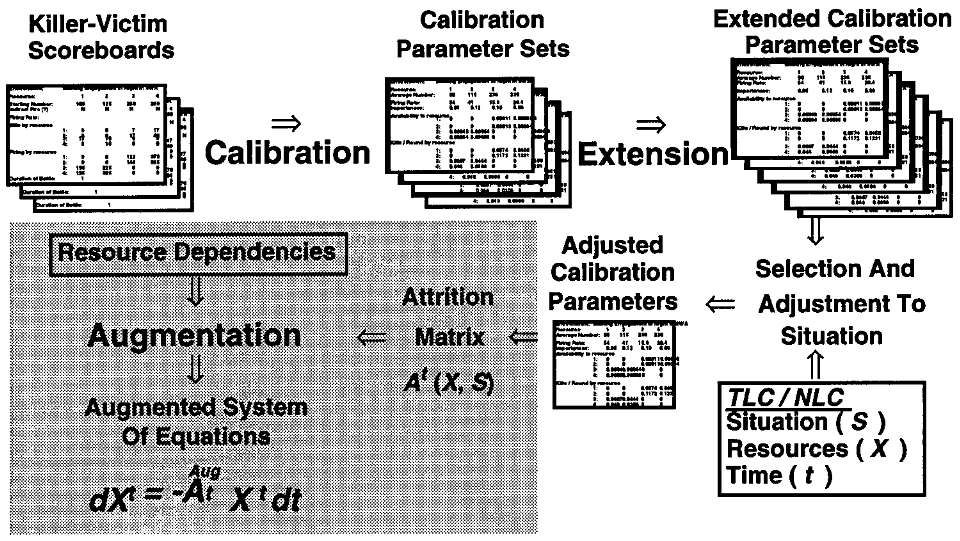

Killer-Victim Scoreboards
Calibration
Calibration Parameter Sets
Extension
Extended Calibration Parameter Sets
Selection And Adjustment To Situation
TLC / NLC
Situation (S)
Resources (X)
Time (t)
Adjusted Calibration Parameters
Attrition Matrix
$A^t(X, S)$
Resource Dependencies
Augmentation
Augmented System Of Equations
$dX^t = -A_t^{Aug} X^t dt$

Figure A.15—Augmentation Step

very tight system of equations, and extra equations cannot be easily added. An open system allows augmentation of equations.

When might the analyst want to augment the equations? Close air support provides an example. In aggregate terms, close air support is a long-term process. Aircraft fly into the battle zone, attack, fly away, return to attack again, and so forth. While the individual attacks are short, the aggregate process is not. The losses they cause can be represented by differential equations. For example, every airplane allocated to the battle might account for one tank per sortie, and an aircraft sorties twice per day. That attrition can be entered into CADEM as a differential equation by augmenting the original system of equations with one that subtracts tanks at the rate of two per day per aircraft.

Recall, however, that importances were calculated based on killing or the degree of threat posed to the enemy by the system. If the resources being added have no killing capability, they will have zero computed importance. They may get killed as a by-product of the battle, but they really do not enter into the importance computation. The calculation for a resource's importance rests not only on the killing it can do but also on its contribution to the killing done by other types of resources. The following equation, using the notation developed earlier, shows the change necessary to the original importance equation to reflect this sort of effect:

$$I_s^3 = \sum_{j=1}^{r}\left[\left(\frac{\partial(\Delta N_j)}{\partial N_s}\Delta t\right)^3\left(F_{s,j}I_j\Delta t \Big/ N_j\Delta N_j\right)\right], s = 1 \ldots r \qquad \text{(A.3)}$$

An example will help clarify the point. It has to do with the dependence of shooters on targeting systems. Some resources do no killing on their own but contribute to the killing process. Artillery forward observers are a case in point. They do not kill anyone usually, but they contribute to the process by directing the fire of artillery to targets.

Figure A.10 assumes two forces composed only of shooters and targeting systems. Without targeting systems, shooters have zero effectiveness. Artillery without any sort of targeting system would have no—or almost no—effect on a battle because it would not know where to shoot. The function in Figure A.16 indicates that effectiveness increases until three targeting systems are present, and no improvement occurs with more. This curve could result from the analysis of several runs in a high-resolution model. This curve can be represented by a differential equation, and a differential equation can be entered as an effect in CADEM.

Table A.5 contains part of the calibration parameter set for this example. The zero values for the targeting systems indicate they do no killing. A calibration step using our original definition of importance would produce the result that the targeting systems have zero importance.

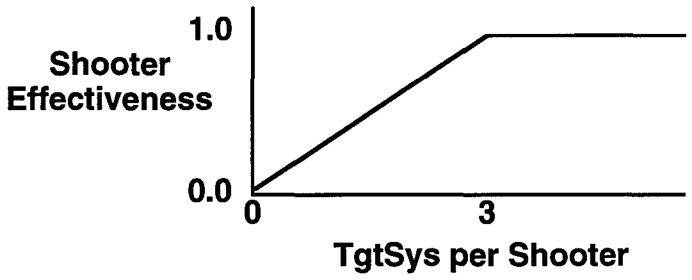


Figure A.16—Dependence of Shooters on Targeting Systems

Table A.5

Shooter/Target System Calibration Set

Type of Battle:	Shooter/TgtSys Example			
For Resource:	1 RED Shooter	2 RED TgtSys	3 BLUE Shooter	4 BLUE TgtSys
Firing Rate:	110	0	21	0
Kills Per Round of 1:			0.45	0.0082
Kills Per Round of 2:			0	0
Kills Per Round of 3:	0.069	0.25		
Kills Per Round of 4:	0	0		
Availability to 1:			.00012	.00035
Availability to 2:			0	0
Availability to 3:	.0045	.0062		
Availability to 4:	0	0		

Figure A.17 graphically depicts the effects of Red targeting systems with and without the augmented definition of importance. The left side shows their importance before augmentation, literally nil. The right side of the chart shows their importance after augmentation. In fact, they do have importance, and, as a result, the importance of Red shooters declines.

Figure A.18 reflects the effects on attrition of the discussed augmentation. The solid lines indicate before augmentation and the dotted lines after. The top two curves depict Red targeting systems. Before augmentation, their attrition is very low. That they have any attrition at all results from the fact that shooters will shoot at a target of no value if they have nothing else to shoot at. However, once augmented, many more Red targeting systems die. Interestingly enough, after augmentation, fewer Red shooters die. Since the importance of Red targeting systems has increased and they are easier to kill than Red shooters, Blue has focused more upon them.

Augmentation could serve a number of other purposes. It might be used to represent multiple weapon systems per vehicle, commitment of reserves, high-value munitions or vehicles, or continuous exogenous attrition (close air support, which would provide attrition over time as opposed to a single catastrophic event, such as a nuclear weapon).

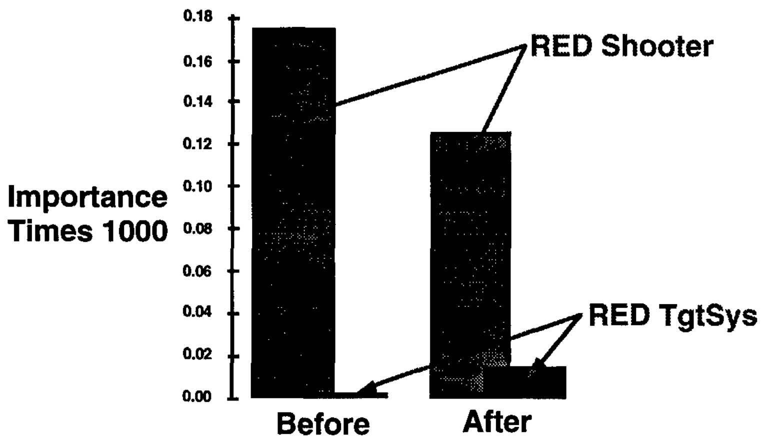


Figure A.17—The Effect of Augmentation on Importance

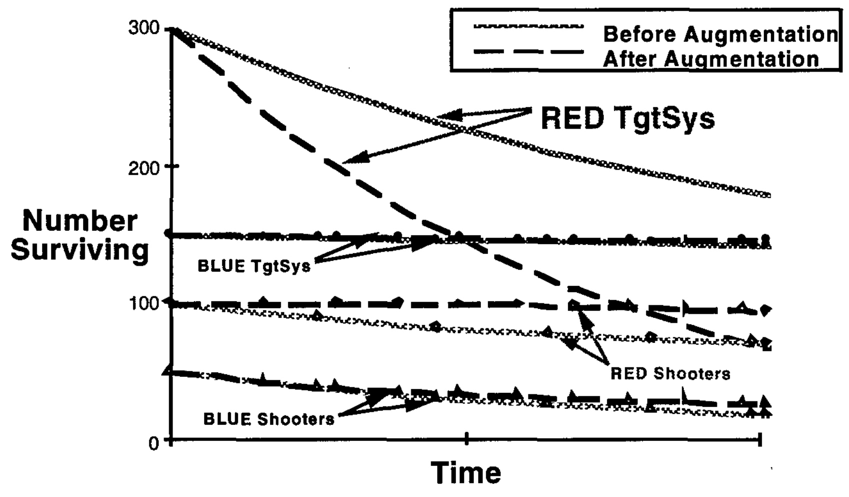


Figure A.18—Augmentation Affects Attrition

Conclusions

The previous subsections have discussed the CADEM process, beginning with killer-victim scoreboards, producing calibration sets, extending those sets, selecting from those sets in a specific situation to calculate an attrition matrix, augmenting that matrix with additional differential equations to represent other facets of the mode, and then calculating surviving resources over time. We conclude by repeating some of the advantages and disadvantages briefly mentioned in the body of this report.

Advantages

The CADEM approach offers a number of advantages over the available aggregate attrition methodologies. Foremost is that it links to high-resolution models and maintains a heterogeneous representation of resources during battle. Its formulation and solution techniques are extensible, that is, capable of being modified. Its applicability may be extended to situations in which data are not available or are inadequate. The system of differential equations may be augmented to capture relationships not adequately captured in killer-victim scoreboards. Further, it offers a spreadsheet implementation that will permit analysts to vary aspects of the model easily and transparently. This feature allows them to get some sense of the effect of various modifications.

Disadvantages

The use of CADEM can have drawbacks. It requires considerable data for specific situations, and these are frequently difficult to get. Developing these and maintaining the collection may be an expensive and time-consuming chore. The ability to augment and extend the process helps to offset the shortage of databases, but it also allows an uninformed analyst to inject unreasonable representations into it. The spreadsheet feature was designed to point out those sorts of errors. In addition, CADEM may take longer to adjudicate combat than other, simpler methodologies. If proper care is taken, this disadvantage may be minimized. CADEM requires less than 50 percent more time than ATCAL to adjudicate the same battle over the same time interval, although CADEM will maintain much better accuracy in some situations.

B. The SAGE Algorithm for Adaptive Resource Allocation

Introduction

This appendix describes an algorithm and some of its variations useful for finding solutions to resource allocation in a conflict model. The basic algorithm attempts to find a saddle point solution using a discrete "min-max" principle much like that of the maximum principle in the field of optimal control. Variations in the algorithm can deal with the complications of mixed or randomized strategies. The primary advantage of the algorithm is that it is "forward searching" and examines only a small region of the state space at any iteration. Thus, the "curse of dimensionality," which prevents the use of dynamic programming in most realistic conflict problems, does not prevent the application of the algorithm. Specific problems based on the allocation of aircraft of two opposing sides to alternative missions to "win" a conflict are used to illustrate the algorithm.

Statement of the Problem and Definition of the Variables

The problem is to find a sequence of strategies for the allocation of resources of two opposing sides in a conflict. The objective of each side is to "win" the conflict on some measure of merit. This may take the form of targets or resources destroyed, final position of the forces, force ratio at the "end" of the conflict, etc. It will be assumed that the objective is zero sum; that is, one side's gain is the other side's loss and vice versa. In an early article, Berkowitz and Dresher (1959) formulated this type of problem for the allocation of 1 and 2 aircraft types and found general solutions for a simple conflict model using dynamic programming. Lloyd Shapley later corrected some of these solutions.[1] TAC CONTENDER (U.S. Air Force, 1971) was an early attempt to solve the same type of problem with a more elaborate model of conflict. It derived the average worth of aircraft and used the resulting values to control the allocation of aircraft to missions. Although it permitted solutions to conflict games with rather large state spaces

[1]Shapley documented these corrections in an unpublished working note.

(that is, state spaces with high dimensionality), it was criticized for its use of expected values to represent initial states at each stage and for its lack of treatment of mixed strategies. DYGAM, by Bracken (1973a, b), attempted to solve the same problem by using an approximate dynamic programming approach. It used a state space grid and polynomial approximations to the optimal value function of dynamic programming to find an optimal mixed strategy. It was criticized because of its limited capability to handle high dimensionality in state space and the fact that states arising from mixed strategies do not necessarily line up with a fixed grid (Dresher, 1975). The approach presented in this paper most closely resembles the TAC CONTENDER work but is based on a discretized version of differential game theory (Freidman, 1971), a field of study most common in optimal control studies. It will be shown that the SAGE approach is also strongly related to differential dynamic programming and other quasi-linearization techniques of control theory (Jacobson and Mayne, 1970; Sage and White, 1977).

State Transition Equations and Variables

It will be assumed that the vector[2] $\mathbf{x}^k$ represents the state of the resources in the conflict at stage k and that knowing this state and the force allocation vectors, $\mathbf{u}^k$ and $\mathbf{v}^k$, of each side (which will be denoted Blue and Red, respectively), permits the complete specification of the state at the beginning of the next stage, $\mathbf{x}^{k+1}$. The change in state from one stage (or period) to the next is given by the vector state transition function, ϕ^k, where

$$\mathbf{x}^{k+1} = \phi^k\left(\mathbf{x}^k, \mathbf{u}^k, \mathbf{v}^k\right). \tag{1}$$

For example, the aircraft of a given type available at the beginning of stage k + 1 of a conflict might be represented by the vector component x_i^{k+1} and given by the state transition equation

$$x_i^{k+1} = x_i^k - L_i\left(\mathbf{x}^k, \mathbf{u}^k, \mathbf{v}^k\right) + R_i^k\left(\mathbf{x}^k, \mathbf{u}^k, \mathbf{v}^k\right), \tag{2}$$

where $L_i\left(\mathbf{x}^k, \mathbf{u}^k, \mathbf{v}^k\right)$ defines the losses of this type of aircraft as a function of the initial state of the system and the allocations in period k and $R_i^k\left(\mathbf{x}^k, \mathbf{u}^k, \mathbf{v}^k\right)$ defines the replacements of this type of aircraft as a function of the state and allocation variables.

[2]Vectors will be indicated by bold type in this appendix.

Objective Function

An objective or objectives can take many forms. A sum of functions of the state and control variables, while not the most general form, seems to be capable of characterizing many objectives of the conflict situations. Let $f_k(\mathbf{x}^k, \mathbf{u}^k, \mathbf{v}^k)$ be the objective value for period k. Then, if the stages or periods are numbered from 1 to N, where N is the final period of optimization, the total objective is given by the expression

$$F^N = \sum_{k=1}^{N} f_k(\mathbf{x}^k, \mathbf{u}^k, \mathbf{v}^k). \tag{3}$$

Frequently, one is also interested in a function of the final state $\mathbf{x}^{N+1}$, which will be represented by $\theta(\mathbf{x}^{N+1})$. The total objective function is then given by

$$O = F^N + \theta(\mathbf{x}^{N+1}). \tag{4}$$

It will be useful to write partial sums of the stage objectives. For this purpose, we define F^K and $\overline{F}^k$ as

$$F^K = \sum_{i=1}^{K} f_i(\mathbf{x}^i, \mathbf{u}^i, \mathbf{v}^i) \tag{5}$$

and

$$\overline{F}^K = \sum_{i=K+1}^{N} f_i(\mathbf{x}^i, \mathbf{u}^i, \mathbf{v}^i). \tag{6}$$

In conflict involving aircraft, f_i may represent targets attacked or destroyed in period i, and the function θ may represent the value of aircraft surviving the N-stage conflict.

The Optimization Problem

Given a fixed number of stages, N, the general problem to be solved is

$$\underset{\mathbf{u}\,\varepsilon\,\mathbf{U}}{\text{minimize}}\ \underset{\mathbf{v}\,\varepsilon\,\mathbf{V}}{\text{maximize}}\ O \tag{7}$$

subject to: $\mathbf{g}(\mathbf{x}, \mathbf{u}, \mathbf{v}) = 0$

where

$\mathbf{u} = \left(\mathbf{u}^1, \mathbf{u}^2, \ldots, \mathbf{u}^N\right)$	is the vector of Blue control vectors across stages. Each $\mathbf{u}^k$ is itself a vector composed of $u_1^k, u_2^k, \ldots, u_I^k$.
$\mathbf{v} = \left(\mathbf{v}^1, \mathbf{v}^2, \ldots, \mathbf{v}^N\right)$	is the vector of Red control vectors across stages. Each $\mathbf{v}^k$ is itself a vector composed of $v_1^k, v_2^k, \ldots, v_J^k$.
$\mathbf{U} = \left(\mathbf{U}^1 \times \mathbf{U}^2 \times \ldots \times \mathbf{U}^N\right)$	is the set of permissible Blue controls. $\mathbf{U}^k$ represents the set of feasible Blue controls in period k. We will assume that $\mathbf{U}^i \cap \mathbf{U}^j = \varnothing$ for all i and j, $i \neq j$. That is, the *feasible* set of controls in one period is not dependent on the feasible set of controls of a previous period.
$\mathbf{V} = \left(\mathbf{V}^1 \times \mathbf{V}^2 \times \ldots \times \mathbf{V}^N\right)$	is the set of permissible Red controls. $\mathbf{V}^k$ represents the set of feasible Red controls in period k. We will assume that $\mathbf{V}^i \cap \mathbf{V}^j = \varnothing$ for all i and j, $i \neq j$.
$\mathbf{g}(\mathbf{x}, \mathbf{u}, \mathbf{v}) = \left(\mathbf{g}^1, \mathbf{g}^2, \ldots, \mathbf{g}^N\right)$	is the vector of state transition constraints. $\mathbf{g}^k$ represents the vector of state transition equations for period k. Thus, g_i^k is a component of $\mathbf{g}^k$ and is given by, $g_i^k\left(x^k, x^{k+1}, u^k, v^k\right) = x_i^{k+1} - \phi_i^k\left(x^k, u^k, v^k\right)$. Note that ϕ_i^k is a component of ϕ^k.

The set of feasible controls in a single period may take the following form in the aircraft conflict model.

$$U^k = \left\{ u^k \,\middle|\, \sum_j u_{ij}^k = 1, u_{ij} \geq 0 \text{ for all aircraft types i, and j is the mission index} \right\}.$$

In this case, the controls represent the mission allocation for an aircraft type, and the constraint set merely represents the fact that the aircraft of each type must be allocated to some mission. In general, even for the aircraft allocation problem, the constraint set is more complicated. There may be constraints that dictate that a minimum or maximum number of aircraft be allocated to a specific mission, and the constraints might be across several aircraft types, indicating the minimum or maximum number of aircraft of all types that can be assigned to a certain mission.

It will be assumed here that the initial state vector, $\mathbf{x}^1$, is known. Modifications to handle a variable initial state can be made. The number of components making up the state vector in each period will be denoted L; the number of components making up the Blue control vector in each period will be I; and the number of components making up the Red control vector in each period will be J. There is no requirement that these be the same numbers for each period, but for ease of exposition, it will be assumed that these limits do not depend on k, the period or stage index.

Derivation of the Algorithm from Necessary Optimality Conditions

The first step in the derivation is to introduce a Lagrange multiplier vector associated with the state transition constraint set to move those constraints to the objective. As will be seen, this serves the highly useful purpose of making the problem separable by the stages. The modified problem is then

$$\underset{\mathbf{u}\,\varepsilon\,\mathbf{U}}{\text{minimize}}\ \underset{\mathbf{v}\,\varepsilon\,\mathbf{V}}{\text{maximize}}\ \{O - \lambda \mathbf{g}(\mathbf{x}, \mathbf{u}, \mathbf{v})\} \tag{8}$$

or

$$\underset{\mathbf{u}\,\varepsilon\,\mathbf{U}}{\text{minimize}}\ \underset{\mathbf{v}\,\varepsilon\,\mathbf{V}}{\text{maximize}} \left\{ \theta\left(\mathbf{x}^{N+1}\right) + \sum_{k=1}^{N} \left\{ f_k\left(\mathbf{x}^k, \mathbf{u}^k, \mathbf{v}^k\right) - \lambda^k \mathbf{g}^k\left(\mathbf{x}^k, \mathbf{x}^{k+1}, \mathbf{u}^k, \mathbf{v}^k\right) \right\} \right\} \tag{9}$$

or

$$\underset{\mathbf{u}\,\varepsilon\,\mathbf{U}}{\min}\ \underset{\mathbf{v}\,\varepsilon\,\mathbf{V}}{\max} \left\{ \theta\left(\mathbf{x}^{N+1}\right) + \sum_{k=1}^{N} \left\{ f_k\left(\mathbf{x}^k, \mathbf{u}^k, \mathbf{v}^k\right) - \lambda^k\left(\mathbf{x}^{k+1} - \phi^k\left(\mathbf{x}^k, \mathbf{u}^k, \mathbf{v}^k\right)\right) \right\} \right\}. \tag{10}$$

Because the control variables and control constraints are assumed to be separable by stage, this problem can be rewritten as

$$\sum_{k=1}^{N} \left\{ \underset{\mathbf{u}^k \varepsilon\, \mathbf{U}^k}{\min}\ \underset{\mathbf{v}^k\, \varepsilon\, \mathbf{V}^k}{\max}\ f_k\left(\mathbf{x}^k, \mathbf{u}^k, \mathbf{v}^k\right) - \lambda^k \mathbf{x}^{k+1} + \lambda^k \phi^k\left(\mathbf{x}^k, \mathbf{u}^k, \mathbf{v}^k\right) \right\} + \theta\left(\mathbf{x}^{N+1}\right). \tag{11}$$

At this point, the discrete Hamiltonian function of optimal control theory can be introduced to simplify the notation:

$$H^k\left(\mathbf{x}^k, \mathbf{u}^k, \mathbf{v}^k\right) = f_k\left(\mathbf{x}^k, \mathbf{u}^k, \mathbf{v}^k\right) + \lambda^k \phi^k\left(\mathbf{x}^k, \mathbf{u}^k, \mathbf{v}^k\right). \tag{12}$$

The optimization problem can now be written as

$$\sum_{k=1}^{N}\left\{\underset{\mathbf{u}^k \varepsilon \mathbf{U}^k}{\text{minimize}}\ \underset{\mathbf{v}^k \varepsilon \mathbf{V}^k}{\text{maximize}}\ H^k\left(\mathbf{x}^k,\mathbf{u}^k,\mathbf{v}^k\right)-\lambda^k\mathbf{x}^{k+1}\right\}+\theta\left(\mathbf{x}^{N+1}\right). \tag{13}$$

Since $\mathbf{x}^{N+1}$ and $\mathbf{x}^{k+1}$ are not variables of the individual stage optimization problems, they can be dropped from the stage objectives. (Recall, this is only because the introduction of the Lagrange multipliers unlinked the stage problems.) The reduced optimization problem is then

$$\sum_{k=1}^{N}\underset{\mathbf{u}^k \varepsilon \mathbf{U}^k}{\text{minimize}}\ \underset{\mathbf{v}^k \varepsilon \mathbf{V}^k}{\text{maximize}}\ H^k\left(\mathbf{x}^k,\mathbf{u}^k,\mathbf{v}^k\right). \tag{14}$$

Now suppose that $\overline{\mathbf{x}},\overline{\mathbf{u}},\overline{\mathbf{v}}$ are optimal solution vectors for this problem. Also assume that the Hamiltonian is differentiable in **x**. It is then necessary for optimality, since (13) is unconstrained in **x**, to satisfy the following:

$$\nabla_x\left(H^k\left(\overline{\mathbf{x}}^k,\overline{\mathbf{u}}^k,\overline{\mathbf{v}}^k\right)-\lambda^{k-1}\overline{\mathbf{x}}^k\right)=0 \text{ for } k=1,\ldots,N. \tag{15}$$

Or

$$\frac{\partial}{\partial x_i}\left(H^k\left(\overline{\mathbf{x}}^k,\overline{\mathbf{u}}^k,\overline{\mathbf{v}}^k\right)-\lambda^{k-1}\overline{\mathbf{x}}^k\right)=0 \text{ for } k=1,\ldots,N \text{ for } i=1,\ldots,L \tag{16}$$

and

$$\frac{\partial}{\partial x_i^{N+1}}\left(\theta\left(\overline{\mathbf{x}}^{N+1}\right)-\lambda^N\overline{\mathbf{x}}^{N+1}\right)=0 \text{ for } i=1,\ldots,L. \tag{17}$$

For $\overline{\mathbf{x}},\overline{\mathbf{u}},\overline{\mathbf{v}}$ to be optimal, it is therefore necessary to find a vector $\overline{\lambda}$ that satisfies the equations

$$\overline{\lambda}_i^{k-1}=\frac{\partial}{\partial x_i^k}H^k\left(\overline{\mathbf{x}}^k,\overline{\mathbf{u}}^k,\overline{\mathbf{v}}^k\right) \text{ for } k=1,\ldots,N \text{ and } i=1,\ldots,L. \tag{18}$$

The initial Lagrange multiplier, λ_i^0, will not be required in the basic algorithm but is used when the initial state is part of the optimization process. The remaining optimality conditions are

$$\overline{\lambda}_i^N=\frac{\partial}{\partial x_i^{N+1}}\theta\left(\overline{\mathbf{x}}^{N+1}\right) \text{ for } i=1,\ldots,L. \tag{19}$$

Of course, it is also necessary that each $\overline{\mathbf{u}}^k$ and $\overline{\mathbf{v}}^k$ solves the optimization problem,

$$\min_{\mathbf{u}^k \varepsilon \mathbf{U}^k} \max_{\mathbf{v}^k \varepsilon \mathbf{V}^k} H^k\left(\overline{\mathbf{x}}^k, \overline{\mathbf{u}}^k, \overline{\mathbf{v}}^k\right) \text{ for } k = 1, \ldots, N \tag{20}$$

and that each $\overline{\mathbf{x}}^k, \overline{\mathbf{x}}^{k+1}$ satisfies the vector state transition equation,

$$\overline{\mathbf{x}}^{k+1} = \phi^k\left(\overline{\mathbf{x}}^k, \overline{\mathbf{u}}^k, \overline{\mathbf{v}}^k\right) . \tag{21}$$

Equations 18 through 21 form the basis of the SAGE algorithm. The multistage objective is conveniently split into a sequence of single-stage optimization problems, and the necessary conditions for optimality provide a convenient method for calculation of the Lagrange multipliers. The algorithm will next be formally stated, its implementation described, and a brief example given.

The SAGE Algorithm

The introduction of the Lagrange multiplier vector to separate the stages creates a discrete two-point boundary value problem much like the continuous two-point boundary problem created by the introduction of the costate vector in optimal control problems (and in differential game theory). The usual approach to solving such problems is to solve for the state vector forward in time (k increasing) and to solve for the Lagrange vector backwards in time (k decreasing). The following algorithm takes the same approach.

1. Let υ be the iteration counter and let $\upsilon = 1$. Let $\mathbf{x}^1(\upsilon) = \overline{\mathbf{x}}^1$ be the given initial state of the system. Assign an initial value to λ^k for each k, say $\lambda^k(\upsilon)$. This value can generally be 0, but it is better from the point of view of convergence to start as close to $\overline{\lambda}^k$ (which is, however, not known) as possible. Let $O(\upsilon)$ be the objective value at iteration υ. Set $O(\upsilon) = 0$. Let ε be an input optimality criterion. Let k be the stage counter. Set k = 1. Go to step 2.

2. Solve the single-stage optimization problem

$$\min_{\mathbf{u}^k \varepsilon \mathbf{U}^k} \max_{\mathbf{v}^k \varepsilon \mathbf{V}^k} H^k\left(\mathbf{x}^k(\upsilon), \mathbf{u}^k, \mathbf{v}^k\right)$$

where

$$H^k\left(\mathbf{x}^k(\upsilon), \mathbf{u}^k, \mathbf{v}^k\right) = f_k\left(\mathbf{x}^k(\upsilon), \mathbf{u}^k, \mathbf{v}^k\right) + \lambda^k(\upsilon)\phi^k\left(\mathbf{x}^k(\upsilon), \mathbf{u}^k, \mathbf{v}^k\right) .$$

Let the resulting solution be $\mathbf{u}^k(\upsilon), \mathbf{v}^k(\upsilon)$. Set $\mathbf{x}^{k+1}(\upsilon) = \phi\left(\mathbf{x}^k(\upsilon), \mathbf{u}^k(\upsilon), \mathbf{v}^k(\upsilon)\right)$. Let $O(\upsilon) = O(\upsilon) + f_k\left(\mathbf{x}^k(\upsilon), \mathbf{u}^k(\upsilon), \mathbf{v}^k(\upsilon)\right)$. Go to step 3.

3. If k = N go to step 4. Otherwise, increment k by 1 and repeat step 2.

4. Set the objective value to $O(\upsilon) = O(\upsilon) + \theta(\mathbf{x}^r(\upsilon))$. If $|O(\upsilon) - O(\upsilon - 1)| < \varepsilon$, then stop under the assumption that the current solution is ε optimal. Otherwise, increment υ by 1 and go to step 5.

5. At this point, the stage counter has value N, and the solution has been determined not to satisfy the ε optimality criterion; hence the Lagrange multipliers must be updated and the problem re-solved. Equations 18 and 19 from the previous subsection define the update. Thus, for k = N backward to 1, we solve

$$\lambda^{k-1}(\upsilon) = \frac{\partial}{\partial x_i^k} H^k\left(\mathbf{x}^k(\upsilon), \mathbf{u}^k(\upsilon), \mathbf{v}^k(\upsilon)\right) \text{ for } i = 1, \ldots, L$$

and

$$\lambda_i^N(\upsilon) = \frac{\partial}{\partial x_i^{N+1}} \theta\left(\mathbf{x}^{N+1}(\upsilon)\right) \text{ for } i = 1, \ldots, L\ .$$

The reader may note that it is not necessary to solve these equations backward in k as they are expressed here. However, this will provide some efficiency and possible faster convergence when the state transition equations take a certain form. Note that $\lambda^k(\upsilon)$, used in the kth stage optimization, cannot be determined until the k + 1st stage optimal solution is in hand. At this point, set the stage counter, k, to 1, go to step 2, and repeat the sequence of period optimizations.

An intuitive feel for how this algorithm solves the multistage optimization problem can be obtained by relating it to the dynamic programming approach to solving the same problem. In dynamic programming, one starts with the last stage and with the optimal value of the terminal state as a function of that state. If we let $O^{N+1}\left(\mathbf{x}^{N+1}\right)$ represent this optimal value as a function of $\mathbf{x}^{N+1}$, then we know that

$$O^{N+1}\left(\mathbf{x}^{N+1}\right) = \theta\left(\mathbf{x}^{N+1}\right). \tag{22}$$

Next, let $O^k\left(\mathbf{x}^k\right)$ be the optimal value from the kth through the final stages as a function of the state at the beginning of the kth stage. At the kth stage, we then can write

$$O^k\left(\mathbf{x}^k\right) = \min_{\mathbf{u}^k \varepsilon \mathbf{U}^k} \max_{\mathbf{v}^k \varepsilon \mathbf{V}^k} \left\{f_k\left(\mathbf{x}^k, \mathbf{u}^k, \mathbf{v}^k\right) + O^{k+1}\left(\mathbf{x}^{k+1}\right)\right\}, \tag{23}$$

with the state transition condition $\mathbf{x}^{k+1} = \phi^k\left(\mathbf{x}^k, \mathbf{u}^k, \mathbf{v}^k\right)$. The process requires this optimization problem to be solved for each possible state at the beginning of stage k. This must be repeated for each of the stages from N backward to 1. The major drawback to this approach for this type of problem is the large number of states that must be evaluated at each stage. For example, if each component, x_i^k, represented the number of aircraft of a given type and there were 10 types of aircraft on each side of a conflict each with 10 possible states, the number of optimization problems to be solved at a single stage could be the prohibitive number 10^{20}. On the other hand, suppose that we know a value $\hat{x}^k$, which is "reasonably" close to the optimal value $\bar{x}^k$. Then, assuming differentiability of $O^k(x^k)$, one could write the linear approximation of this function about $\hat{x}^k$ as

$$O^k(x^k) \cong O^k(\hat{x}^k) + \nabla_x O^k(\hat{x}^k)(x^k - \hat{x}^k). \tag{24}$$

Using this linearization for $O^{k+1}(x^{k+1})$, we could then write the kth stage dynamic program as

$$\begin{aligned} O^k(x^k) \cong \min_{u^k \varepsilon U^k} \max_{v^k \varepsilon V^k} \Big\{ & f_k(x^k, u^k, v^k) + O^{k+1}(\hat{x}^{k+1}) \\ & + \nabla_x O^{k+1}(\hat{x}^{k+1})(x^{k+1} - \hat{x}^{k+1})\Big\}. \end{aligned} \tag{25}$$

Next, let $\nabla_x O^{k+1}(\hat{x}^{k+1}) = \xi^k$ and substitute the state expression, $x^{k+1} = \phi^k(x^k, u^k, v^k)$. The resulting optimization problem is

$$\begin{aligned} O^k(x^k) \cong \min_{u^k \varepsilon U^k} \max_{v^k \varepsilon V^k} \Big\{ & f_k(x^k, u^k, v^k) + O^{k+1}(\hat{x}^{k+1}) \\ & + \xi^k\big(\phi^k(x^k, u^k, v^k) - \hat{x}^{k+1}\big)\Big\}. \end{aligned} \tag{26}$$

The constant terms in the objective function can be removed, and the following approximate dynamic programming problem must be solved at each stage:

$$\min_{u^k \varepsilon U^k} \max_{v^k \varepsilon V^k} \Big\{ f_k(x^k, u^k, v^k) + \xi^k\big(\phi^k(x^k, u^k, v^k)\big)\Big\}. \tag{27}$$

Thus, through the process of linearization, it can be seen that the dynamic programming problem looks quite similar to the problem that is solved at each stage of the SAGE algorithm. To show that the two problems are really the same, it must be shown that ξ^k, the vector obtained by differentiating the optimal value function of dynamic programming, is the same as λ^k, the vector obtained by differentiating H^{k+1}, the Hamiltonian of the next stage. To see this, note that

$$\lambda^{k-1} = \nabla_x H^k(x^k, u^k, v^k) \tag{28}$$

and

$$\xi^{k-1} = \nabla_x O^k\left(\mathbf{x}^k\right). \tag{29}$$

Now, starting with λ^N and ξ^N,

$$\lambda^N = \nabla_x \theta\left(\mathbf{x}^{N+1}\right) \tag{30}$$

and

$$\xi^N = \nabla_x \theta\left(\mathbf{x}^{N+1}\right). \tag{31}$$

Next, at stage N – 1,

$$\lambda^{N-1} = \nabla_x H^N\left(\mathbf{x}^N\right) = \nabla_x\left\{f_N\left(\mathbf{x}^N, \mathbf{u}^N, \mathbf{v}^N\right) + \lambda^N\left(\phi^N\left(\mathbf{x}^N, \mathbf{u}^N, \mathbf{v}^N\right)\right)\right\} \tag{32}$$

and

$$\begin{aligned} \xi^{N-1} = \nabla_x O^N\left(\hat{\mathbf{x}}^N\right) &= \nabla_x\left\{f_N\left(\mathbf{x}^N, \mathbf{u}^N, \mathbf{v}^N\right) + \xi^N\left(\phi^N\left(\mathbf{x}^N, \mathbf{u}^N, \mathbf{v}^N\right)\right)\right. \\ &\quad \left. -\xi^N\hat{\mathbf{x}}^N + O^N\left(\hat{\mathbf{x}}^N\right)\right\} \\ &= \nabla_x\left\{f_N\left(\mathbf{x}^N, \mathbf{u}^N, \mathbf{v}^N\right) + \xi^N\left(\phi^N\left(\mathbf{x}^N, \mathbf{u}^N, \mathbf{v}^N\right)\right)\right\} \end{aligned} \tag{33}$$

Thus, by looking at the recursive generation of λ^k and ξ^k, one can see that they are the same and that the dynamic programming problem and SAGE algorithm solve the same sequence of optimization problems. The SAGE algorithm solves the multistage problem by what amounts to a piecewise linearization of the optimal value function of dynamic programming. The advantage of the SAGE algorithm is that it arrives at the linearization point, $\hat{\mathbf{x}}$, by an iterative, fixed-point approach and does not require the combinatorial evaluation of optimization problems at each stage as a function of a large number of states. Of course, one must be concerned about the algorithm's convergence to the proper fixed point from a possibly arbitrary starting point. Convergence will be discussed later in this appendix. The important point is that this approach has the potential for solving large multistage game problems that would be intractable by normal methods.

Solution of a Simple Conflict Problem with the SAGE Algorithm

The following problem was chosen to illustrate the algorithm, rather than to represent the true complexities of aircraft allocation in a conflict situation. This problem was first described and solved as part of the TAC CONTENDER work (U.S. Air Force, 1971). Assume that Blue and Red each have only one type of aircraft and that these aircraft are capable of doing two types of missions, attacking the air bases of the other side (to destroy aircraft) and attacking the ground forces of the other side (to destroy land forces). Assume that the objective of each side is to maximize its destruction of the other side's land forces and to minimize the destruction of its own land forces by the opposing air force. Each side will also place some positive value on the number of its aircraft surviving and a negative value on the number of the opponent's aircraft surviving. Let $\mathbf{x}^k$ represent the number of aircraft available at the beginning of stage k. That is, x_b^k will be the state variable giving the number of Blue aircraft at the beginning of stage k, and x_r^k will be the state variable giving the number of Red aircraft at the beginning of stage k. Let $\mathbf{u}^k$ and $\mathbf{v}^k$ represent the Blue and Red control vectors, respectively, at stage k. Each control vector will have two components, the fraction of aircraft allocated to attacking ground forces and the fraction allocated to attacking air bases. These components will be designated u_g^k, v_g^k and u_a^k, v_a^k respectively. The aircraft must all be allocated to one mission or the other so that the control constraints are

$$\mathbf{U}^k\left\{u_g^k, u_a^k \middle| u_g^k + u_a^k = 1, u_g^k \geq 0, u_a^k \geq 0\right\} \tag{34}$$

and

$$\mathbf{V}^k\left\{v_g^k, v_a^k \middle| v_g^k + v_a^k = 1, v_g^k \geq 0, v_a^k \geq 0\right\}. \tag{35}$$

When an air base is attacked, the number of aircraft destroyed will be assumed proportional to the number of attacking aircraft. This is limited by the number of aircraft of the opposing side. The simplifying assumption will be made that this is equal to all of the aircraft available at the beginning of the stage. Similarly, the quantity of ground forces destroyed in an attack on land forces will be proportional to the number of attacking aircraft. The state transition equations are then

$$x_b^{k+1} = \phi_b^k\left(\mathbf{x}^k, \mathbf{u}^k, \mathbf{v}^k\right) = x_b^k - \text{minimum}\left(d_a^r v_a^k x_r^k, x_b^k\right) \tag{36}$$

and

$$x_r^{k+1} = \phi_r^k\left(\mathbf{x}^k, \mathbf{u}^k, \mathbf{v}^k\right) = x_r^k - \text{minimum}\left(d_a^b u_a^k x_b^k, x_r^k\right)\ , \tag{37}$$

where d_a^r and d_a^b are the number of aircraft destroyed on air bases per Red and Blue aircraft assigned to attack air bases. We will assume for purposes of this illustration that neither side attacks the other side's air bases with enough force to destroy all of the aircraft. The state transition expressions then simplify to

$$x_b^{k+1} = x_b^k - d_a^r v_a^k x_r^k \tag{38}$$

and

$$x_r^{k+1} = x_r^k - d_a^b u_a^k x_b^k \ . \tag{39}$$

The objective function is given by

$$\begin{aligned} O = F^N + \theta\left(x^{N+1}\right) &= \sum_{k=1}^{N} f_k\left(x^k, u^k, v^k\right) + \theta\left(x^{N+1}\right) \\ &= \sum_{k=1}^{N}\left(d_g^b u_g^k x_b^k - d_g^r v_g^k x_r^k\right) + \left(w_b x_b^N - w_r x_r^N\right) , \end{aligned} \tag{40}$$

where d_g^b and d_g^r are the quantities of land force targets destroyed per Blue and Red aircraft allocated to attacking land force targets and where w_b and w_r represent the values of Blue and Red aircraft surviving N stages of conflict, respectively. Finally, the initial numbers of Blue and Red aircraft, x_b^1 and x_r^1, will be assumed to be known quantities. The problem, then, is to find the optimal sequence of Blue and Red aircraft allocations such that Blue maximizes and Red minimizes the given objective function. The Hamiltonian for each stage is

$$\begin{aligned} H^k\left(x^k, u^k, v^k\right) = \left(d_g^b u_g^k x_b^k - d_g^r v_g^k x_r^k\right) + \lambda_b^k\left(x_b^k - d_a^r v_a^k x_r^k\right) \\ - \lambda_r^k\left(x_r^k - d_a^b u_a^k x_b^k\right). \end{aligned} \tag{41}$$

The signs of λ_b^k and λ_r^k are arbitrary (since they are associated with an equality constraint). Therefore, they have been chosen to have opposite signs in the Hamiltonian expression to achieve symmetry in the expressions below.

The Hamiltonian optimization problem for each stage is

$$\begin{aligned} \min_{u_g^k, U_a^k} \ \max_{v_g^k \cdot V_a^k} \ \Big\{\left(d_g^b u_g^k x_b^k - d_g^r v_g^k x_r^k\right) + \lambda_b^k\left(x_b^k - d_a^r v_a^k x_r^k\right) \\ -\lambda_r^k\left(x_r^k - d_a^b u_a^k x_b^k\right)\Big\} , \end{aligned} \tag{42}$$

subject to

$$u_g^k + u_a^k = 1, u_g^k \geq 0, u_a^k \geq 0$$

and

$$v_g^k + v_a^k = 1, v_g^k \geq 0, v_a^k \geq 0 \ .$$

The following solution to this linear programming problem should be quickly obvious:

Set $u_g^k = 1, u_a^k = 0$ if $d_g^b \geq \lambda_r^k d_a^b$. Otherwise set $u_g^k = 0, u_a^k = 1$.

Set $v_g^k = 1, v_a^k = 0$ if $d_g^r \geq \lambda_b^k d_a^r$. Otherwise set $v_g^k = 0, v_a^k = 1$.

Basically, this solution indicates that the strategy of a side should be to attack land forces when the marginal payoff of the attack per aircraft is greater than the marginal payoff of attacking the other side's air bases per aircraft. The latter payoff is dependent on the Lagrange multiplier associated with the other side's aircraft. The effect of the Lagrange multiplier is to give the marginal value of the opponent's aircraft. The multipliers are updated after the initial guess, using

$$\lambda_b^{k-1} = \frac{\partial H^k}{\partial x_b^k} = d_g^b u_g^k + \lambda_r^k d_a^b u_a^k \tag{43}$$

and

$$\lambda_r^{k-1} = \frac{\partial H^k}{\partial x_r^k} = d_g^r v_g^k + \lambda_r^k + \lambda_b^k d_a^r v_a^k \ . \tag{44}$$

The algorithm would proceed with a given initial set of λ^k and x^1, solve the sequence of stage-optimization problems, update the Lagrange multipliers, re-solve the stage problems, etc. Table B.1 gives the results of three iterations of a four-period problem with $d_g^b = d_g^r = 1.0$, $d_a^b = 0.6, d_a^r = 0.4, x_b^1 = 100, x_r^1 = 100$ and $w_b = w_r = 0$. Note that the optimal solution requires Blue to use all of its aircraft to attack Red air bases for 2 days (stages) and then switch to attacking land forces for the next 2 days. Red must attack Blue's air bases for 1 day and then attack land-force targets. Initial Lagrange multipliers were arbitrarily chosen to have values of 0 for each side for each day of iteration 1. At iterations 2 and 3, these were updated using equations (43) and (44). Note that, by starting at period N and working backward to find λ^{N-1}, etc., one could use the updated values, $\lambda^k(\upsilon)$, rather than $\lambda^k(\upsilon - 1)$ to determine $\lambda^{k-1}(\upsilon)$. In practice, this seems to speed convergence of the algorithm.

The fact that the solution is arrived at in so few iterations is due to the simplicity and linearity of this example. Empirically, for more complex conflict situations, the required number of iterations to achieve "reasonably" stable strategies seems to be roughly equal to the number of periods or stages.

In this example, each side allocated its entire air force to one mission or the other at each stage. This occurs in this example because the objective is linear and the

Table B.1

Iterations to Solve a Simple Two Sided Aircraft Conflict Problem

υ	k	x_b^k	x_r^k	u_g^k	u_a^k	v_g^k	v_a^k	λ_b^k	λ_r^k	F^k
1	1	100	1	0	1	0	0	0	0	0
1	2	100	100	1	0	1	0	0	0	0
1	3	100	100	1	0	1	0	0	0	0
1	4	100	100	1	0	1	0	0	0	0
2	1	100	100	0	1	0	1	3	3	0
2	2	60	40	0	1	1	0	2	2	–40
2	3	60	4	1	0	1	0	1	1	16
2	4	60	4	1	0	1	0	0	0	72
3	1	100	100	0	1	0	1	32	3	0
3	2	60	40	0	1	1	0	2	2	–40
3	3	60	4	1	0	1	0	1	1	16
3	4	60	4	1	0	1	0	0	0	72

constraints permit such an allocation. In general, the conflict objective is likely to be nonlinear (it is nonlinear even in this example when the limit on number of aircraft that can be destroyed is imposed as in equations [36] and [37]). This can lead to partial allocations among different missions. Another type of partial allocation occurs when there is no single Blue and Red strategy pair that satisfies equation (19) at some stage, k, in step 2 of the algorithm. This requires a mixed strategy in which probabilities of choosing strategies are introduced. We have developed the algorithm to find mixed strategies, but the lengthy mathematical description of the approach will not be presented in this paper.

Our experience with the SAGE algorithm has been limited to the study of allocation of tactical aircraft to missions in central conflict. In these applications, we have considered problems of fairly high dimensionality (up to 20 types of aircraft on each side with up to five missions to allocate for each). To date, we have found that convergence has been quite rapid—a number of iterations roughly equal to the number of periods, depending, of course, on the tightness of the convergence criteria. Furthermore, except for relatively simple, specially constructed cases, the solutions have endured the test of detailed scrutiny with respect to their optimality. That is, it has not been possible to find better solutions or determine that they are nonoptimal local solutions based on expert opinion and perturbation analysis. A further curious, but welcome, outcome has been the relative infrequency of mixed strategy solutions. While this may be a result of the particular problem parameters, it may also be a result of the high dimensionality. Perhaps with a large number of state variables and allocation possibilities on each side, the existence of pure, stable equilibria is more likely.

There are many unexplored extensions and applications of this type of algorithm. There is no theoretical reason why the initial state of the system to be examined cannot be made a part of the solution. For example, it may be of interest to ascertain the appropriate number of aircraft of various types to move to a theater of conflict prior to the initiation of hostilities. This implies that cost functions for the mobilization be included in the formulation and that these functions be optimized with the conflict objective. It may also be possible to consider a time objective much in the way that control theory deals with minimum time problems. For example, the goal of conflict might be to minimax the time to reach a given objective. Another interesting but possibly quite difficult problem is to include some aspects of parameter uncertainty, much in the way optimal control of "noisy" processes is achieved. Clearly, if some of these problems can also be solved, the applications of this type of algorithm to conflict and competitive situations are much wider.

Bibliography

Allen, Patrick D., *Situational Force Scoring: Accounting for Combined Arms Effects in Aggregate Combat Models*, Santa Monica, Calif: RAND, N-3423-NA, 1992.

Analysis Support Directorate, *Attrition Calibration (ATCAL) Evaluation Phase I—Direct Fire (ATVAL PHASE I)*, Bethesda, Md., CAA-SR-91-10, July 1991.

Analysis Support Directorate, *ATCAL: An Attrition Model Using Calibrated Parameters*, Bethesda, Md., CAA-TP-83-3, August 1983.

Anderson, Lowell Bruce, and Frederick A. Mieroth, Institute for Defense Analysis, "On Weapon Scores and Free Strength," in Jerome Bracken, Moshe Kriss, and Richard E. Rosenthal, eds., MORS, *Warfare Modeling*, Danvar, Md.: John Wiley and Sons Inc., 1995, pp. 199–228.

Bailey, Timothy, et al., *Mobile Strike Force 95 Organizational and Operational Analysis*, Ft. Leavenworth, Kan.: U.S. Army TRADOC Analysis Center, September 1995.

Bennett, Bruce, "Report on Panel Two: Representing Military Activities," in R. Hillestad, R. Huber, and M. Weiner, *New Issues and Tools for Future Military Analysis: A Workshop Summary*, Santa Monica, Calif: RAND, N-3403-DARPA/AF/A, 1992.

_____, *JICM 1.0 Summary*, Santa Monica, Calif: RAND, MR-383-NA, 1994.

Bennett, Bruce, et al., *RSAS 4.6 Summary*, Santa Monica, Calif: RAND, N-3534-NA, 1993.

Berkowitz, D., and M. Dresher, "A Game-Theory Analysis of Tactical Air War," *Operations Research*, Vol. 17, 1959.

Bracken, Jerome, *Two Optimal Sortie Allocation Models*, Vol. I: *Methodology and Sample Results*, Arlington, Va.: Institute for Defense Analyses, AD 771 779, December 1973a.

_____, *Two Optimal Sortie Allocation Models*, Vol. II: *Computer Program Documentation*, Arlington, Va.: Institute for Defense Analyses, AD 771 779, December 1973b.

CACI Products Company, *MODSIM II, The Language for Object-Oriented Programming: Reference Manual*, La Jolla, Calif., May 1991.

_____, *TAC THUNDER Analytical Model and War Game: A Theater-Level Combat Simulation*, Arlington, Va., n.d.

Chu, David, "World Change and Military Operations Research: The View from OSD," *Phalanx*, September 1991, pp. 1, 10–15.

Commission on Roles and Missions of the Armed Forces, "Direction for Defense," *Report of the Commission on Roles and Missions of the Armed Forces*, John White, Chairman, May 24, 1995.

Davis, Paul, and Donald Blumenthal, *The Base of Sand Problem: A White Paper on the State of Military Combat Modeling*, Santa Monica, Calif: RAND, N-3148-OSD/DARPA, 1991.

Davis, Paul, and Reiner Huber, *Variable-Resolution Combat Modeling: Motivations, Issues, and Principles*, Santa Monica, Calif: RAND, N-3400-DARPA, 1992.

Davis, Paul K., and Richard J. Hillestad, "Families of Models That Cross Levels of Resolution: Issues for Design, Calibration and Management," in G. W. Evans, M. Mollaghasemi, E. C. Russell, and W. E. Biles, eds., *Proceedings of the 1993 Winter Simulation Conference*, 1993.

Decision-Science Applications, *The TAC BRAWLER Air Combat Simulation Analyst Manual* (Revision 5.0), Washington, D. C., DSA Report 906, June 1988.

Department of the Air Force, *Air Force Modeling and Simulation: A New Vector*, 1995.

Department of the Army, *U.S. Army Field Manual*, Washington, D.C., FM 100-5, 14 June 1993.

Department of Defense, *Joint Munitions Effectiveness Manual* (U), n.d. Classified publication, not for public release.

Deutch, John M., Deputy Secretary of the Air Force, Memorandum for the Director, Program Analysis and Evaluation, on Joint Theater Models, February 8, 1995.

DIS Steering Committee, *The DIS Vision: A Map to the Future of Distributed Simulation*, Version I, May 1994.

Dresher, Melvin, *The N-Stage Game and Lagrangian Multipliers: The N-Stage Game and DYGAM*, Naval Warfare Research Center, Technical Note 61, November 1975.

Dupuy, Trevor, *Understanding War: History and a Theory of Combat*, New York: Paragon House Publishers, 1987.

Fishman, George S., *Principles of Discrete Event Simulation*, New York: John Wiley & Sons, 1978.

Freidman, Avner, *Differential Games*, New York: Wiley-Interscience, 1971.

Harrison, George B., "An Open Letter to the Analytical Community," *Phalanx*, December 1991, pp. 1–3.

Hillestad, Richard, J., *Dyna-METRIC: Dynamic Multi-Echelon Technique for Recoverable Item Control*, Santa Monica, Calif: RAND, R-2785-AF, July 1982.

Hillestad, Richard J., and Manuel J. Carrillo, *Models and Techniques for Recoverable Item Stockage When Demand and the Repair Process Are Nonstationary*, Part 1: *Performance Measurement*, Santa Monica, Calif: RAND, N-1482-AF, 1980.

Hillestad, R., R. Huber, and M. Weiner, *New Issues and Tools for Future Military Analysis: A Workshop Summary*, Santa Monica, Calif: RAND, N-3403-DARPA/AF/A, 1992.

Hillestad, Richard J., and Mario L. Juncosa, "Cutting Some Trees to See the Forest: On Aggregation and Disaggregation in Combat Models," *Naval Research Logistics*, Vol. 22, 1995, pp. 183–208.

Hillestad, R. J., M. S. Weiner, and E. L. Warner, *Allied Air Forces Central Europe Air Campaign Study* (U), Santa Monica, Calif: RAND, R-3897-AF, 1992. Classified publication, not for public release.

Hillestad, Richard, and John Owen, "Experiments in Variable Resolution Modeling," *Naval Research Logistics*, Vol. 42, 1995, pp. 209–232.

Hines, John G., "Calculating War, Calculating Peace: Soviet Military Determinants of Sufficiency in Europe," in Reiner Huber, ed., *Military Stability: Prerequisites and Analysis Requirements for Conventional Stability in Europe*, Baden-Baden, FRG: Nomos-Verlag, 1990, pp. 185–199.

Hollis, Walter, W. "Responding to the Fluid Influences Facing the Army and Its Analysts," *Phalanx*, September 1991, pp. 1, 7–9.

Huber, Reiner, "An Analytical Approach to Cooperative Security: First Order Assessment of Force Posture Requirements in a Changing Security Environment," in Reiner Huber, ed., *Military Stability: Prerequisites and Analysis Requirements for Conventional Stability in Europe*, Baden-Baden, FRG: Nomos-Verlag, 1990, pp. 165–184.

Hughes, Wayne P., Jr., ed., *Military Modeling*, 2nd. ed., Alexandria, Va.: Military Operations Research Society, 1989.

Jacobson, David H., and David W. Mayne, *Differential Dynamic Programming*, New York: American Elsevier Publishing Company, 1970.

Jefferson, D. R., and H. Sowizral, *Fast Concurrent Simulations Using the Time Warp Mechanism*, Part I: *Local Control*, Santa Monica, Calif: RAND, N-1906-AF, December 1982.

Kiviat, P. J., R. Villanueva, and H. M. Markowitz, eds., *SIMSCRIPT II.5 Programming Language*, 4th ed., Los Angeles, Calif.: CACI Products Company, 1984.

Louer, Phillip E., "Army Combat Simulation Assessment," *Phalanx*, December 1991, pp. 7–10.

Marti, Jed, and Barbara Gates, "An Empirical Study of Time Warp Request Mechanisms," *Distributed Simulation*, Vol. 19, No. 3, 1988.

Marti, Jed, and C. Burdorf, "Non-Preemptive Time Warp Scheduling Algorithms," *Operating Systems Review*, Vol. 24, No. 2, April 1990, pp. 7–18.

Marti, Jed, P. Kantar, W. Sollfrey, and K. Brendley, *SEMINT: Seamless Model Integration*, Santa Monica, Calif: RAND, MR-403-OSD/A, 1994.

McDonough, Larry, Scott Bailey, and Allison Koehler, *MapView User's Guide*, Santa Monica, Calif: RAND, MR-160-AF/A, 1993.

Members of the M1A2 SIMNET-D Synthetic Environment Experiment Committee, *Results of the M1A2 SIMNET-D Synthetic Environment Post-Experiment Analysis*, May 1993.

Olsen, P., U. Candan, and J. J. de Jijs, *IDAHEX Version 4*, Vol. III: *Player's Manual*, The Hague, the Netherlands: SHAPE Technical Centre, STC TM-763, File ref. 9980, 1985.

Parker, T. M., and L. Wegner, *Model and Data for Interdiction Analysis* (U), R-3743/4-AF, Santa Monica, Calif: RAND, August 1989. Classified publication, not for public release.

Press, William H., et al., *Numerical Recipes: The Art of Scientific Computing*, Cambridge, Mass.: Cambridge University Press, 1989.

Pyles, Raymond, *The DYNA-METRIC Readiness Assessment Model: Motivation, Capabilities, and Use*, Santa Monica, Calif: RAND, R-2886-AF, 1984.

R&D Associates, *Battle Command Training Program (BCTP) Opposing Force (OPFOR) Command and Staff Handbook*, Book 2, Chapter Five, 15 July 1990, pp. 5.89–5.96.

Repass, Todd H., *Special Report: ADS/DIS*, "DIS: An Evolving Modus Operandi for the Department of the Navy," n.d.

Sage, Andrew P., and Chelsea C. White III, *Optimum Systems Control*, 2nd. ed., Englewood Cliffs, N.J.: Prentice Hall, Inc., 1977.

Stein, Charles, "A Two-Sample Test for a Linear Hypothesis Whose Power is Independent of the Variance," *Annals of Mathematical Statistics*, Vol. 16, 1945, pp. 243–258.

Taylor, James G., *Lanchester Models of Warfare*, Vol. I, Arlington, Va.: Operations Research Society of America, 1983.

Teledyne Brown Engineering, *EADSIM Methodology Manual, Version 4.0*, March 30, 1994.

Tragemann, Richard W., "Analysis Trends in the 90's and Beyond," *Phalanx*, September 1991, pp. 25–27.

U.S. Air Force, Assistant Chief of Staff, Studies and Analysis, *Methodology for Use in Measuring the Effectiveness of General Purpose Forces (An Algorithm for Approximating the Game Theoretic Value of N-Staged Games)*, Saber Grand (Alpha), March 1971.

U.S. Army, Concepts Analysis Agency, *Concepts Evaluation Model VI (CEM VI)*, Vol. I: Technical Description, Bethesda, Md., October 1987.

U.S. Army, Concepts Analysis Agency, *Concepts Evaluation Model VII (CEM VII)*, Vol. II: User's Handbook, CAA-D-85-1, Bethesda, Md., December 1991.

U.S. Army, TACWAR Project Officer, *TACWAR Ground Analyst Guide*, Model Version 4.0, Fort Leavenworth, Kan., July 1994a.

U.S. Army, TRADOC, *Force XXI Operations: A Concept for the Evolution of Full-Dimensional Operations for the Strategic Army of the Early Twenty-First Century*, Pamphlet 525-5, August 1, 1994b.

U.S. Department of Commerce, National Technical Information Service, *World Data Bank II*, 1977.