AD-A060 402 DECISION-SCIENCE APPLICATIONS INC ARLINGTON VA F/G 12/2
VALUE-DRIVEN DECISION THEORY: APPLICATION TO COMBAT SIMULATIONS--ETC(U)
JUL 78 G L LUCAS, G F GORMAN, G E PUGH F49620-77-C-0089
UNCLASSIFIED DSA-67 AFOSR-TR-78-1398 NL

1 OF 3
AD A060 402

78-1398

(2) LEVEL II

Report No. DSA-67

See back page for 1473

VALUE-DRIVEN DECISION THEORY

APPLICATION TO COMBAT SIMULATIONS

G. L. Lucas
G. F. Gorman
G. E. Pugh

July 1978

prepared for:

Director, Mathematical and Information Science
AIR FORCE OFFICE OF SCIENTIFIC RESEARCH
Building 410, Bolling AFB, D.C. 20332

78 10 10 111

DECISION-SCIENCE APPLICATIONS, INC.
1500 WILSON BOULEVARD, SUITE 810, ARLINGTON, VIRGINIA 22209

Research sponsored by the Air Force Office of Scientific Research (AFSC), United States Air Force, under contract F49620-77-C-0089. The United States Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation hereon.

Report No. DSA-67

VALUE-DRIVEN DECISION THEORY

APPLICATION TO COMBAT SIMULATIONS

G. L. Lucas
G. F. Gorman
G. E. Pugh

July 1978

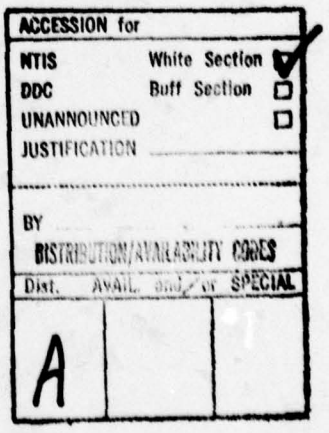
prepared for:

Director, Mathematical and Information Science
AIR FORCE OFFICE OF SCIENTIFIC RESEARCH
Building 410, Bolling AFB, D.C. 20332

DECISION-SCIENCE APPLICATIONS, INC.
1500 WILSON BOULEVARD, SUITE 810, ARLINGTON, VIRGINIA 22209

ACKNOWLEDGMENT

The authors would like to thank the following members of the DSA support staff for contributing substantively to the preparation of this report: Maree Clark, Marilee Seastrand, and Victoria Haynes.



CONTENTS

FIGURES

FIGURES (CONT'D.)

TABLES

I. AN OVERVIEW OF VALUE-DRIVEN DECISION THEORY

A. INTRODUCTION

The proper modeling of command, control, and information (C^2I), as it affects combat performance, has been one of the most difficult problems confronting the combat simulation designer. This problem has become particularly acute in recent years because of the need to assess the combat effectiveness of major advances in both sensor systems and information processing. Although it is recognized that combat performance depends critically on the availability of timely and relevant information, the lack of procedures for quantifying the implications of improved information flow has made it extremely difficult to assess the combat performance of new weapons systems. For example, improved information has no effect on the maneuverability of a particular aircraft or the rate of fire of a particular gun; however, it can profoundly influence combat outcomes by changing the choice of missions for the aircraft or the aimpoint for the gun.

To represent the effect of information quality on combat outcomes, it is necessary to model the way that combat decisions are influenced by the availability of information. Until recently, no effective modeling procedures have been available for realistically representing combat decision processes in computer simulations. Recent theoretical developments in the understanding of human decision processes, however, appear to offer the possibility of realistically simulating command-and-control processes. The new approach that is used to model the effects of C^2I is described as an information-oriented, and value-driven, simulation. This type of combat model simulates not only the physical interactions between combatants, but also the effects of information that is used by combatants to make decisions in response to a changing combat environment.

Decision-making in a value-driven simulation follows a procedure that parallels the human decision process. A decision element considers a number of alternative courses of action and uses an imperfect model of the real world (its mental model) to project probable outcomes for the alternatives. The mental model reflects both the decision element's knowledge of and uncertainty about the combat environment. Like real combatants, a simulated combatant projects outcomes only a short distance into the future and re-examines his decisions periodically as the situation evolves. Because outcomes are projected only a short time into the future, the simulated combatant must necessarily use heuristic value criteria to evaluate the projected outcomes. To make the simulation perform as realistically as possible, the criteria are adjusted to correspond as closely as possible to the heuristic values that might be used by a human decision-maker in a similar situation.

Heuristic values also serve as the medium for communicating "preferences" between decision elements. Orders, for example, are communicated between decision elements by modifying the value structure of the receiving element, so that the element tends to act in accordance with the intentions of the sending element. This procedure forms the basis for realistically representing command structures in value-driven simulations.

Value-driven simulations thus provide a useful vehicle not only for studying the effects of improved information flow in combat simulations, but also for developing insights into complex combat operations involving many interacting decision-makers. The method has thus far been fully exploited in the development of many-on-many combat simulations in which the individual pilots are explicitly modeled, and is the design of the C^2I system for the Air Force's Combined Arms Simulation Model (CASM). Noncombat applications are also under development.

The present report was prepared by Decision-Science Applications, Inc., (DSA) under Contract F49620-77-C-0089 to the Air Force Office of Scientific Research. Although intended as a handbook for the design and development of value-driven simulations, the report is not intended as a design "cookbook." Rather, it has been structured to present the basic philosophy for the design of value-driven simulations and the tested techniques necessary to implement them.

This report is organized as follows. Chapter I presents the basic theory of value-driven system design. It is intended as a general overview of the principal concepts for the non-designer. Chapter II describes the design consideration necessary to build a value-driven simulation. Chapters III and IV describe value structure design. Chapter V examines the generation of alternatives. Chapter VI describes the construction of the mental model, and Chapter VII discusses computer implementation concepts in value-driven design.

B. BACKGROUND

The origin of value-driven decision theory can be traced to the work of a small group of scientists at Lambda Corporation in the early 1960's. These scientists were concerned with the development of strategic warfare models for use in evaluating the effectiveness of U.S. strategic forces--missiles and bombers--against the Soviet target base. The principal problem faced by these scientists was the allocation of over 1,000 strategic weapons, some of them MIRVed, against a Soviet target system that comprised in excess of 5,000 targets. Traditional methods for performing such allocations employed laydown procedures, in which values were first ascribed to targets and then weapons assigned to the targets, one at a time, in such a way that the weapon effecting maximum damage was assigned first, the weapon effecting the second-most damage second, and so forth, until the weapon inventory was exhausted. These procedures had the advantage that they were easy to apply and produced reasonable solutions. The solutions were, however, nonoptimal and the degree of nonoptimality, i.e., the difference between the value of the

optimal and the realized solution, could vary substantially from case to case. Since many studies of interest were concerned with relatively small differences in effectiveness between alternative force structures, the laydown procedures were not satisfactory. An optimal method--or at least a method in which the deviation from optimal could be accurately estimated--was needed.

To meet this need, the Lambda scientists developed a method known as General Lagrange Multiplier (GLM) theory.[1] This method differed from ordinary Lagrange Multiplier theory--which can be applied only to differentiable functions defined on a piecewise continuous region--in that it could be applied to an arbitrary bounded function defined on an arbitrary region. Moreover, the method was particularly adaptable to the computer, so that very large problems, such as the strategic assignment problem introduced above, could be readily solved.[2] The GLM method was first applied in the QUICK General Wargaming System where it was used to accomplish the assignment of U.S. and Soviet forces in a nuclear exchange.[3] This system successfully performed the detailed allocation of bombers and land- and sea-launched missiles to targets taking into account numerous real-world considerations usually ignored in computerized models. The system is still used today by the Joint Chiefs of Staff to evaluate the SIOP, the U.S. Strategic Integrated Operational Plan.

---

[1] H. Everett, III, "Generalized Lagrange Multiplier Method for Solving Problems of Optimal Allocation of Resources," Operations Research 11, 397-417 (1963). The mathematical formalism was actually developed by Dr. Everett at the Institute for Defense Analyses (IDA). He subsequently founded Lambda Corporation where the method was developed more fully.

[2] The method is described in detail in Chapter III.

[3] The NMCSSC QUICK-Reacting General War Gaming System (QUICK). National Military Command System Support Center, July 1967.

The QUICK General Wargaming System was a forerunner of the modern value-driven decision system, and it included a number of the features of the modern system: it included a highly sophisticated, computerized decision algorithm, capable of making realistic real-world decisions; it associated values, albeit in highly simplified form, with alternative courses of action and used a value maximization principle to select an operative course of action; and it utilized the concept of Generalized Lagrange Multiplier, which is one of the primary implementation themes of the value-driven method. Nevertheless, the QUICK General Wargaming System lacked many of the features that define the modern value-driven method. Of most significance is the concept of a mental model, which is used by a simulated decision entity to form its own perception of the simulated environment. The decision entity to form its own perception of the simulated environment. The decision entity interprets simulated "sensor" data (in light of rules of information interpretation encoded into the entity); it generates alternative courses of action, associates values with the alternatives (as opposed to having them preassigned), and selects a preferred course of action. These and the other capabilities that now characterize the method evolved over an extended period of time.

The initial refinements in the value-driven method paralleled the advances in the development of the Generalized Lagrange Multiplier method. The first of these was the extension of the GLM to handle two-sided game theoretic and minmax problems.[1] This capability permitted combat scenarios to be represented in situations where the combatants acted in an adversary role, each trying to maximize their advantage over the other. Early applications of the method were to strategic scenarios, in which the defense allocated his defense--generally terminal ABMs--to minimize the damage the offensive would inflict, and the offense--generally with knowledge of the disposition of the defense--allocated his forces to maximize the damage inflicted.

[1] G. Pugh, "Lagrange Multipliers and the Optimal Allocation of Defense Resources," Operations Research, 12, 4 (1964).

The development of a pharmaceutical production scheduling system for Merck & Company represented the next significant advancement in the development of a value-driven method.[1] Merck was exploring a new plant design concept, in which multipurpose equipment, i.e., centrifuges, filters, tanks, etc., could be combined in "erector set" fashion to produce a desired product set. The proposed plant was to produce up to 150 products using over 200 different module types. The essential problem was to develop an automated scheduler that would schedule production over approximately a one-year time horizon in such a way that corporate profit would be maximized and the size of the plant, for a given level of production, would be minimized.

The development of the Merck scheduler introduced three new features to the value-driven method. First, and most simply, it represented the first treatment of dynamic play, in which the method is applied to a system evolving over time. Previous applications, such as in QUICK or in the game theoretical play, addressed only one-time assignments or at most strike-counterstrike exchanges. Second, and much more importantly, the Merck scheduler represented the beginning of the systematic development of heuristic guidelines for value-driven systems. The Merck scheduler was not a rigorous optimization system but rather employed heuristic procedures for obtaining near-optimal solutions. It was recognized in the design of this scheduling system that rigorous optimization is often neither feasible nor desirable in "real-world" operational systems, and so the emphasis in developing value-driven decision systems shifted from a search for optimality to the development of systematic heuristic procedures for representing real-world systems.[2] Third, and finally, the

---

[1] G. Lucas and G. Pugh, A Modernized Plant Design and Scheduling System, Volumes I, II, III, IV, V, Merck & Company, December 1970.

[2] Later, when the emphasis in developing value-driven decision systems focused on the representation of the human decision-maker, it was recognized that strict optimality was also not a characteristic of the human decision-maker, so that heuristic procedures which provide good but not necessarily optimal solutions, often provide the most satisfactory representation of the decision-maker.

Merck scheduler represented the introduction of complex value functions to the value-driven method. In traditional methods, such as linear programming, values are input to the model and the program finds the optimal solution for these predefined values. In the Merck scheduler the values are generated dynamically during the execution of the program, responding to inventory shortages, module availabilities, and current production schedules. The dynamic generation and adaptability of the "values" is a hallmark of the value-driven method. In the scheduler the values are generated and the production decisions made using a modified form of the Generalized Lagrange Multiplier method.

The next step in the development of the value-driven decision method was realized in the 44-City Study, in which student busing plans for 44 metropolitan areas were developed for the Department of Health, Education, and Welfare (HEW).[1] Because of controversy over the busing of school children that had developed as a result of court ordered desegregation plans, HEW was interested in efficient student assignment plans that would minimize the number of busing miles required, subject to constraints on the racial composition and the capacity of the schools and to certain subsidiary constraints on, for example, the maximum distance any student would be bused.

The development of the computerized decision algorithm for generating the student assignment plans introduced two new features to the value-driven method. First, it represented the first time that complex alternative courses of action had to be explicitly generated. In most optimization problems, the feasible courses of action are either readily enumerated, e.g., in dynamic programming, or are automatically generated by the solution method, e.g., in linear programming. By contrast, in the development of the busing plans, specific bus routes had to be constructed (starting from a detailed road map); combined into alternative courses of action;

---

[1] G. E. Pugh and H. Everett, School Desegregation with Minimum Busing, Lambda Paper #68, December 1971.

evaluated; and then compared to other potential courses of action. The necessity for generating alternative courses of action and the guidelines used in generating them constitutes an important feature of the value-driven method. This is particularly true in ground combat applications where there is a clear need for generating penetrations, envelop envelopments, and other complex maneuvers.

A second important feature introduced by the development of the busing plans for HEW concerns the development of value functions for large systems and the treatment of subsidiary constraints. Initial attempts to generate busing plans, taking into account only the major constraints, revealed that there were many solutions (i.e., busing plans) whose values were arbitrarily close to the value of the optimal solutions, and that among these there were invariably some that satisfied the subsidiary constraints. The important question then became how to automatically select the desirable solutions without explicitly taking the subsidiary constraints into account and thereby immensely complicating the analyses. The solution was found to be simply to add any small term to the value function that would discriminate among the degenerate solutions. This was a particularly important observation, for it led to the use of the multiple component value functions that are integral to the present theory, and it led to the realization that the precise form of the value function that is used to characterize the worth of an alternative is not critical, so that standardized forms for the value function could be adopted for use in many practical applications.

The development of a formal and rigorous procedure for characterizing dynamic play in combat games and the subsequent development of heuristic implementation procedures constituted the next step in the development of the value-driven decision method. The mathematical formulation of the dynamic optimization problem, when expressed in the General Lagrange Multiplier formalism, was found to decouple the successive stages of the problem in a simple and intuitive way, permitting the dynamic problem to be rigorously treated as a multistage optimization problem, in which each stage could be independently optimized. The decoupling was effected by

means of the Lagrange Multipliers (or shadow values), which reflected the value of withholding a weapon for future employment. This conceptually simple interpretation of the Lagrange Multiplier permitted the heuristic solution of many problems that were too complex to be rigorously treated by the method. The most well-known application of the method was in the MUSTEX system,[1] a multiple-stage strategic nuclear exchange model which has been extensively employed by the Chief of Naval Operations to study strategic problems.

The first model in which the concept of the mental model as it exists in the modern value-driven decision system arose was TAC COMMANDER, a model developed for the Air Force Assistant Chief of Staff for Studies and Analysis to simulate the tactical air command-and-control system for NATO forces in Western Europe.[2,3] In TAC COMMANDER, reconnaissance aircraft are committed to specified areas to gather information on the activities of the ground forces. This information is then communicated back to a central decision center where it is interpreted, prioritized, assigned to a decision cue, and later used in the assignment of strike aircraft to targets. The interpretation of the data is effected primarily in terms of the perishability of the targets, which is then translated into an urgency to attacks the targets and used in setting target priorities. The targets are then processed according to their priority. The decision process in TAC COMMANDER, which concerns the assignment of aircraft from selected airbases to targets, was originally structured using decision rules, which specified for a given set of conditions the airbase and the aircraft to select for a given strike.

---

[1] D. Noble and G. Pugh, Staged Counterforce Exchanges, 1985-1990: Implications for Strategic Force Composition and Characteristics, GRC Report 904-01-CR, November 1976.

[2] G. Lucas and S. Collier, TAC COMMANDER: A Description of the Intermediate Priority-Driven Model, Lambda Report 152, January 1975.

[3] S. Collier and G. Pugh, TAC COMMANDER: A Description of the Value-Driven System, GRC Report CR-110, May 1975.

This was subsequently changed to a value-driven scheme in which the choice of aircraft was determined by considering both the probable damage to a target and the probability an aircraft might be lost on the mission.

TAC COMMANDER thus possessed many of the features of the modern value-driven decision system. An internal mental model of the external world gave TAC COMMANDER the capability to intercept sensory data and to assess the urgency of assigning strike aircraft to targets. Specific aircraft from special airbases were then assigned, initially according to a series of decision or priority rules which reflected particular tactics hardwired into the model, and later according to a value structure which favored the adoption of particular tactics but would allow deviations from them to account for unusual circumstances. In this revised form, the tactics themselves were subject to modification simply by modifying the parameters characterizing the value functions rather than by modifying the code itself. TAC COMMANDER thus not only assumed the form of the modern value-driven system, but acquired much of its versatility as well.

The FIMOD air defense simulation, developed for the Institute for Defense Analyses (IDA) to provide an effective comparison of various F-15 and F-16 force mixes in the defense of Europe, extended the value-driven assignment concepts developed in TAC COMMANDER to include multiple decision elements.[1] Three decision levels are explicitly represented in FIMOD: External Control, Flight Leader Control, and Pilot Control. Command information is communicated between these levels through a command language appropriate to each level. The use of explicit command channels allows each level to respond automatically to the decision made at the next higher level.

FIMOD therefore extends the concepts developed in TAC COMMANDER to provide the hierarchical control structure necessary for the control of

[1] Value-Driven Simulations and FIMOD, GRC Report IP-01-W, December 1975.

hundreds of aircraft. Use of such a hierarchical control structure make it possible to use a comparatively simple and transparent logic to deal with the decisions at each level, for the decision processes do not have to be concerned with the details of how the decision is implemented at the lower levels. Although the decision logic in FIMOD was primarily concerned with the efficient utilization of interceptor resources (as in TAC COMMANDER), the use of command hierarchies and central linkages formed the basis for inter- and intra-decision element linkages discussed throughout this report.

The full power of the value-driven decision method was not realized, however, until the development of the TAC BRAWLER, a many-on-many tactical aircraft simulation in which individual pilots are explicitly modeled as distinct decision entities.[1] Each decision entity in TAC BRAWLER has access only to data received via its own sensors or via communication links with other aircraft. Data received via these links is integrated into an internal mental model of the external world, which is formed by the integration of previous perceptions received via sensor and command links into a pre-existing structural model of the external world. Employing its revised mental model, each decision maker makes an appraisal of the extant world situation, constructs potentially attractive courses of action, projects their outcomes, associates values or measures of worth with each outcome, and selects a preferred course of action. An action, for example, may be a particular maneuver designed to initiate attack against a hostile aircraft or to evade an attacker, or it may be a communication from a flight leader to a wingman instructing him to engage or to break off an particular contact. The consequence of such

[1] G. Gorman and R. Kerchner, TAC FLIGHT: A Value-Driven Multi-Aircraft Simulation for Analysis of Close Air Combat, GRC Report 913-01-CR, November 1977. (The name of the TAC FLIGHT simulation was later changed to TAC BRAWLER.)

a communication would be a revision in the value function structure of the wingman to increase tbe desirability of pursuing courses of action consistent with the directive of the flight leader.

By the use of such procedures and by the development of a sufficiently robust mental model of the external world, a command-and-control system and the decision elements incumbent to it can be faithfully and precisely represented, and the resultant simulation structure can be used both to reflect the effects of command and control in combat simulations and to study command-and-control systems themselves. In the following section, the basic conceptual structure of the value-driven approach is briefly outlined as it will be developed in the remainder of the report.

C. THE VALUE-DRIVEN PARADIGM

The value-driven decision approach to the modeling of C^3 in combat simulations comprises both a formal structure and a body of guidelines and techniques for use in applying the approach to combat simulations. This section describes the basic logical structure of the value-driven approach. Subsequent sections elaborate on the approach and compare it with other approaches for treating C^3 in combat simulations. The balance of the report is concerned with an exposition of the guidelines and techniques for implementing the value-driven approach in combat simulations.

The essential element of the value-driven approach is the decision element. The formal structure of the decision element is shown in Fig. I.1. The decision element comprises all the essential features and characteristics necessary to represent decision processes in combat simulations. This includes the capability to receive and interpret sensor and communications data, to form an internal mental model of the external world, to generate possible courses of action and to project their consequences, and to select and direct the implementation of a particular course of action.

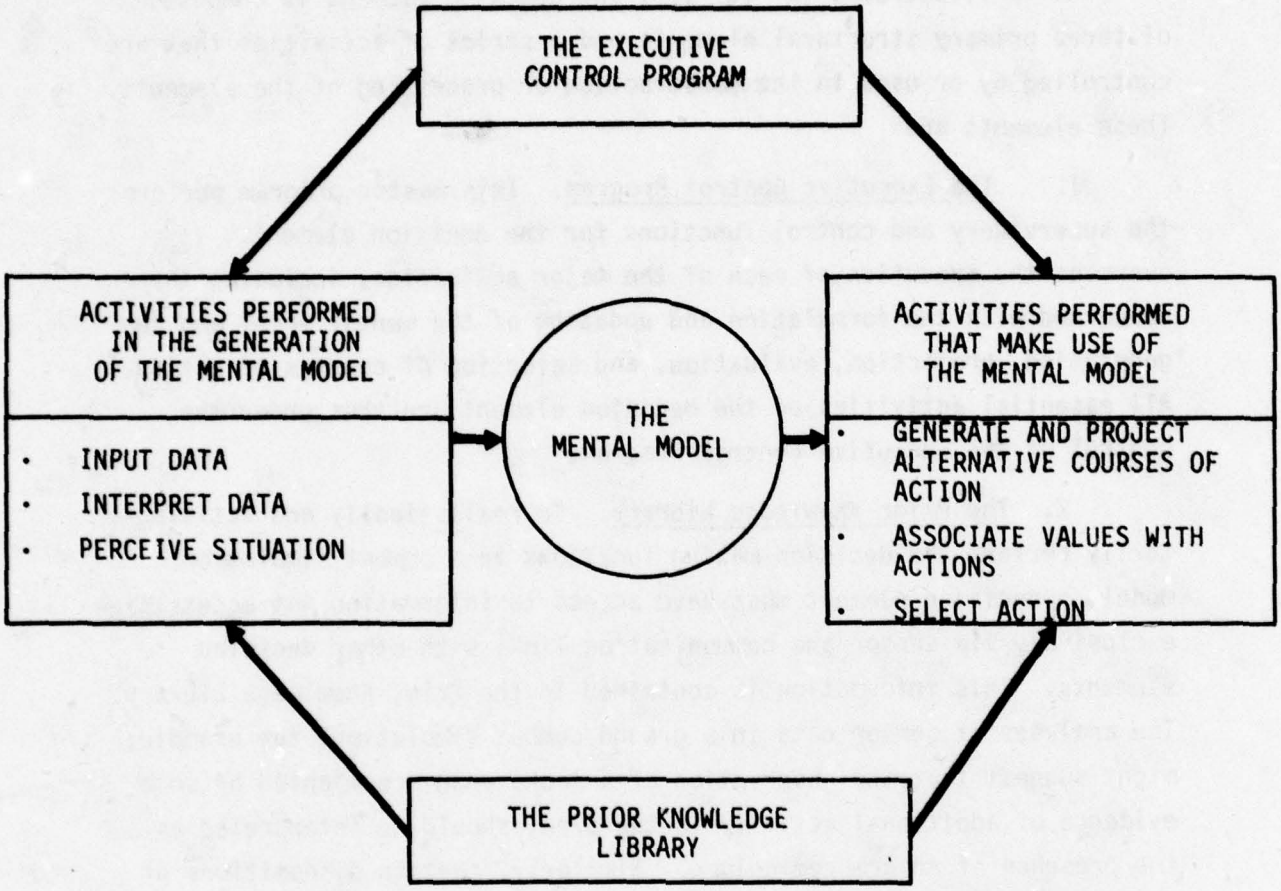


Figure I.1. Logical Structure of the Decision Element

As is illustrated in Fig. I.1, the decision element is composed of three primary structural elements and a series of activities that are controlled by or used in the construction or processing of the elements. These elements are:

1. The Executive Control Program. This master program performs the supervisory and control functions for the decision element. It oversees the execution of each of the major activities, including those concerned with the formulation and updating of the mental model and the generation, projection, evaluation, and selection of courses of action. All essential activities of the decision element are thus under the control of the Executive Control Program.

2. The Prior Knowledge Library. To realistically and satisfactorily perform the decision-making functions in a combat simulation model, a decision element must have access to information not accessible exclusively via sensor and communication links with other decision elements. This information is contained in the Prior Knowledge Library. The analyses of sensor data in a ground combat simulation, for example, might suggest that the observation of a tank, when accompanied by some evidence of additional activity in the area, should be interpreted as the presence of an armored column. Similarly, certain dispositions of hostile forces might suggest the onset of an attack. In these cases, without the preconceived notion of an armored column or without the preconception of "threatening dispositions of forces," the decision element would be unable to correctly diagnose the situation and initiate a proper response. The situation is very similar to that encountered in robotics, where a mobile robot guided by a television camera must interpret a certain configuration of lines as a solid object lying in its path. The robot must then take corrective action to reach its objective. Without the prior knowlege that such configurations of lines correspond to "objects to be avoided" the robot would be unable to initiate the proper response. The prior knowledge of the objects to be avoided must be built into the decision logic of the robot.

Similar considerations apply to the generation, projection, and evaluation, of courses of actions. The type of prior knowledge required here takes three forms. The first is simply the knowledge of the rules or laws of action which permit the decision element to project the consequences of pursuing a given course of action. In the ground combat simulation, for example, a decision element "contemplating" a penetration of the enemy lines must have a basis for evaluating the worth of initiating the action. To carry out this evaluation, the decision elements must have the means of deducing the consequences of the action given its perception of the overall situation.

The second type of prior knowledge concerns the set of alternative courses of action that are generated as candidates for implementation. This is an area that has been the most seriously neglected in the construction of classical campaign models and is a major source of the objection that the models do not faithfully reflect the way "people really fight." In particular, most models do not reflect the asymmetry between the way the NATO and Warsaw Pact forces contemplate fighting in Europe. The second type of preknowledge thus concerns knowledge of the doctrine and tactics that guide the selection by the decision element of promising courses of action.

The final type of prior knowledge concerns the evaluation of the courses of action for the purpose of the selection of an alternative for implementation. Measures or values have to be available to the mental model for determining the relative or, in some circumstances, absolute worth of pursuing a particular course of action. For example, in a ground combat simulation measures must be available for assessing the relative attractiveness of initiating a flanking attack compared to undertaking a broad frontal assault. The primary valuative procedures for discriminating among such alternatives therefore must be part of the prior knowledge available to the decision element.[1]

[1] We temporarily omit adaptive models from consideration. Even these, however, must have ultimate or primary values to guide the adaptive process.

3. The Mental Model. For a decision element to contemplate possible courses of actions, to project their outcomes, and to evaluate their utility requires a "mental picture" or "mental model" of the current state of the external world. This mental model then serves as the basis for all activities of the decision element. In the value-driven decision method the mental model is developed wholly from the prior knowledge available to the decision element and from the data received during the simulation via sensor and communication links with other decision elements. Construction of the mental model is guided by the requirement for generating a perception of those aspects of the external world that are necessary for projecting and evaluating alternative courses of action. Referring once again to the example from robotics, the mobile robot must develop, from the collection of lines presented to it, preceptions corresponding to "objects to be avoided" and also to "areas of potential avoidance," the latter corresponding to regions blocked from the robot's field of view. With these perceptions, the robot has a sufficiently complete mental model of the external world to construct and evaluate alternative routes that are potentially available to it for reaching its objective. Similarly, in a ground combat simulation, a decision element contemplating a potential penetration must have a geographical model of the prospective penetration area, including possible routes and defended terrain, as well as sufficient sensor and communications data on the distribution and strength of friendly and hostile forces to permit it to evalute the feasibility and effectiveness of pursuing such a course of action.

The series of information processing activities engaged in by the decision entity can be conveniently divided into those activities involved in the generation of the mental model and those that use the mental model in the generation, evaluation, and selection,of courses of actions. Those activities that are primarily involved in the generation of the mental model are:

- Input data. This activity consists of the entering of sensor and communication data into the decision element for use in updating the mental model. Commonly included in the activity are the introduction of uncertainties in data, time delays, data loss, incorrect data, and other features reflecting the imperfection of the sensors and other data sources that limit the quality and timeliness of the data available to the decision element.

- Interpret data. This activity consists of the initial and primarily low-level interpretation of sensor (and possibly communication) data. It is most readily defined for the ground combat model where it can be characterized as the construction of a "situation map," consisting of the location--real or surmised--of friendly and hostile forces and, where feasible, indicating their direction of movement and other characteristics which are subject to observation. The type of inferences required for data interpretation is exhibited by the example cited in the discussion of the Prior Knowledge Library, in which the observation of a tank accompanied by some evidence of additional activity suggested the presence of an armored column in the area.

- Perceive situation. This activity consists of the generation or updating of the mental model employing the new and interpreted information received via sensors or communication links with other decision elements. It differs from the data interpretation function, in that it consists not of generating the situation map but of analyzing it to extract key variables that are relevant to the generation of candidate courses of action. By analogy with the human decision process it can be referred to as "action-oriented perception," in that the guiding principle in situation perception is to focus on that portion of the external world that is relevant

to the actions that are available for implementation by the decision element. Situation perception is thus closely and inextricably intertwined with the generation of alternative courses of action. Examples of situation perception range from the assessment of the intentions of hostile aircraft and the determination of hostile aircraft that are promising candidates for attack by friendly aircraft in TAC BRAWLER to the determination of the density of enemy forces for use in estimating the rate of advancement and ultimately the likelihood of successful penetration in a ground combat simulation.

Those activities that may be primarily characterized as using the mental model in the generation and selection of courses of action are:

- Generate and project alternative courses of action. A principal determining feature of the value-driven decision approach is the explicit dynamic generation of alternative courses of action by a decision element during the execution of the simulation. In a ground combat simulation, for example, a division-level decision element may consider several different penetration alternatives and then project the consequences of selecting each alternative as a basis for selecting an alternative for implementation. Similarly, in TAC BRAWLER, a pilot might consider several different maneuvers, project the consequences of selecting each maneuver, and then select for implementation the maneuver that projects to the most desirable state. In each case, for the division level element and for the pilot, the projection of the alternative will normally be carried out using a much simpler and more aggregated model than is used in the actual play of the game, both to more realistically simulate the limitations of the human decision process and to conserve computer resources. This simplification of models used in the projection of alternatives is a general feature of the value-driven method.

- Associate values with alternative courses of action. The association of values with alternative courses of action is the characteristic that gives the value-driven decision method its name. Values provide the mechanism by which alternatives can be compared and by which preferred alternatives can be selected for implementation. They are the principal feature that drives the simulation and the feature for which the greatest effort has been required for generating guidelines and techniques for use in their construction. In addition to their role in alternative selection, they play a fundamental role in the transmission of orders and directives from one level of a command hierarchy to another. In TAC BRAWLER, for example, an order from a flight leader to a wingman to attack a particular hostile aircraft is reflected in the simulation as an enhancement in the wingman's value functions for those alternatives representing an attack on the hostile aircraft.

- Select alternative. This activity consists of the selection by the decision element of a particular course of action. Although employing values to make the selection, the selection process is generally conducted in a manner that differs from that employed in most allocation methods, for the emphasis is on identifying an alternative that provides an acceptable but not necessarily optimal solution. Following the examination of a small but carefully selected set of representative solutions, additional alternatives are examined--generally one by one--until a satisfactory solution is identified or a limit is reached on the number of solutions to be examined. This procedure is adopted both to more closely simulate human decision processes and to provide a mechanism for generating courses of action in situations like penetrations where the actions are difficult to construct and enumerate. The emphasis in this aspect of the design process is on developing

procedures to ensure that the set of solutions examined are sufficiently robust to contain at least one acceptable solution and on developing stopping rules for terminating the search procedure.

The Executive Control Program, the Prior Knowledge Library, the Mental Model, and the five primary activities (input data, interpret data, perceive situation, generate and project courses of action, associate values with actions, and select an alternative) constitute the logical framework and dynamic behavior functions of the decision element. For many practical applications, this structure as described and as exhibited in Fig. I.1 provides an entirely satisfactory representation of the decision processes in combat simulations. For a simulation like TAC BRAWLER, however, in which the individual pilots are explicitly modeled, the spectrum of decisions that must be considered is so diverse and the input requirements so varied that the decision processes are most efficiently represented in the form of a hierarchy of decision elements. Such a "multi-tiered" decision element is shown for TAC BRAWLER in Fig. I.2. The decision element, as displayed, represents the flight leader, who has responsibility both for decisions concerning the activities of the flight as a whole and for pilot-related decisions concerning the activities of his own aircraft. The upper dotted box in the figure corresponds to those decision processes related to flight activities; the lower dotted box to those related to the flight of his own aircraft. A wingman, who does not have command responsibilities for the flight, would be represented by a decision element containing only the lower dotted box.

The TAC BRAWLER multi-tiered decision element illustrates a number of features of the value-driven approach. In addition to the division of the decision element into the two major tiers corresponding to the flight-related and pilot-related decision types, there is a further division of each category into posture and tactical decision levels and, in the lower box, into a subconscious decision level. These levels reflect a hierarchical decision process in which general policy or posture decisions are made at the upper level and tactical or implementation decisions reflecting the upper level decisions are made at the lower level.

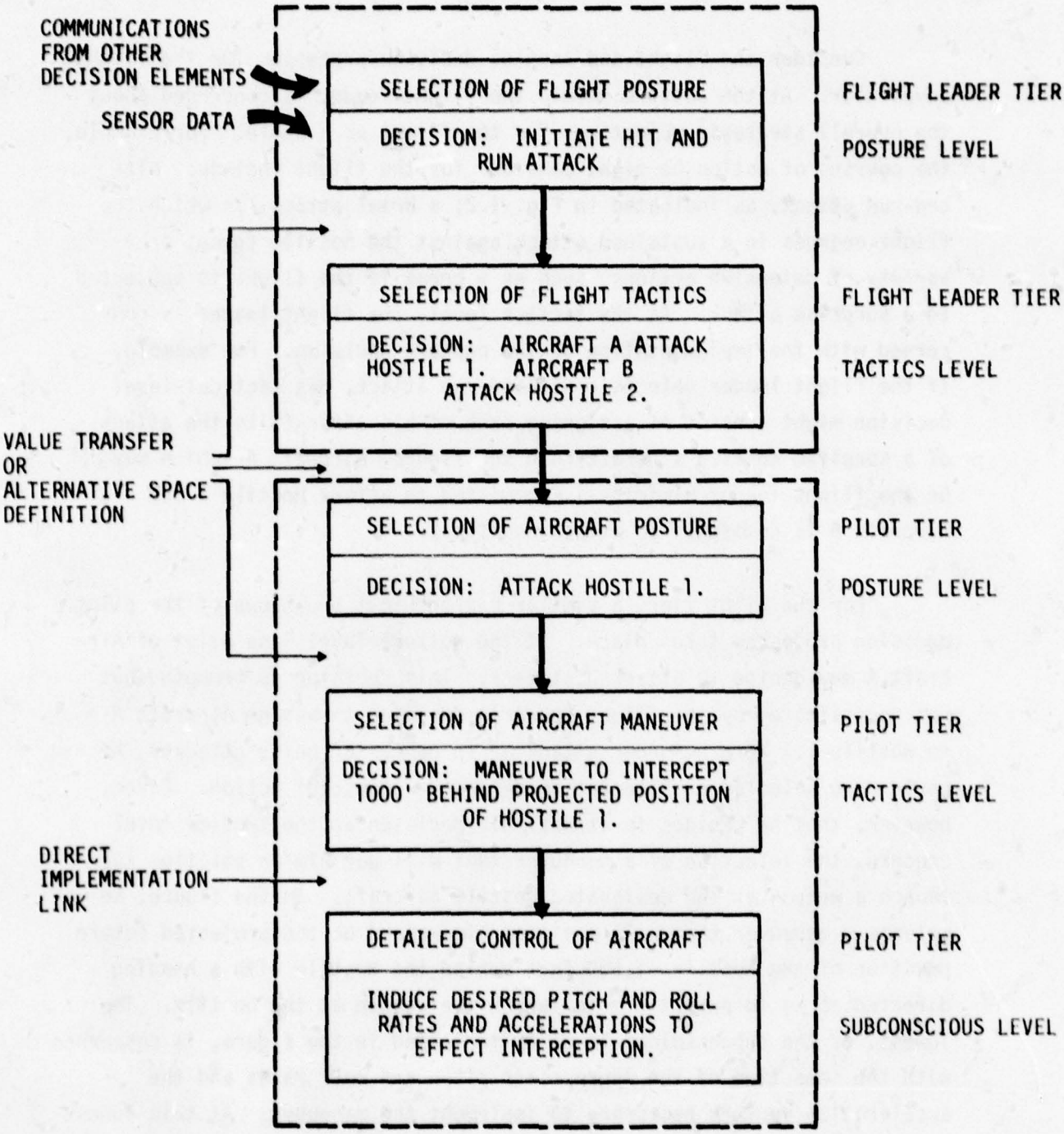


Figure I.2. Multi-Tiered Decision Element for TAC BRAWLER

Consider the flight and tactics decision processes for the flight-level tier. At the posture level, the flight leader is concerned about the overall strategic objectives for the flight as a whole. For example, the courses of action he might consider for the flight include: hit-and-run attack, as indicated in Fig. I.2; a brawl attack, in which the flight engages in a sustained attack against the hostile force; or a variety of defensive actions, such as a break if the flight is subjected to a surprise attack. At the tactics level, the flight leader is concerned with the implementation of the posture decision. For example, if the flight leader selects a hit-and-run attack, his tactical-level decision might consist of assigning each of his aircraft to the attack of a specific hostile aircraft. In the figure, Aircraft A--which may be the flight leader himself--is committed to attack Hostile 1 and Aircraft B is committed to attack Hostile 2.

For the pilot tier, a similar hierarchical breakdown of the pilot's decision processes takes place. At the posture level, the pilot of Aircraft A may decide to attack Hostile 1. This decision is prompted but not necessitated by the flight leader's decision to assign Aircraft A to Hostile 1. Were he under attack or in danger of being attacked, he could have selected an alternative defensive course of action. Given, however, that he decides to attack, his decision at the tactics level concerns the selection of a maneuver that will put him in position to launch a weapon at the designated hostile aircraft. In the figure, he selects a maneuver that will position him--based on the projected future position of the hostile--1,000 feet behind the hostile with a heading directed so as to permit him to launch the weapon at the hostile. The lowest, or the subconscious level as indicated in the figure, is concerned with the selection of the appropriate pitch and roll rates and the acceleration vectors necessary to implement the maneuver. At this lowest level, no explicit decision processes are required and the decision element degenerates into a rule-based selection mechanism, e.g., the rates and acceleration are simply chosen so as to induce the specified maneuver. No generation of alternatives, value associations, and so forth, are required.

Several observations can be made based on examining the TAC BRAWLER multi-tiered decision element. First of all, note that substantially different information is necessary for making the decisions required at each of the decision levels. For example, for the flight leader tier, the information required and the considerations necessary to make the decision to initiate an attack differs substantially from those required to make the detailed assignment of friendly to hostile aircraft. The decision to initiate an attack requires information not only on the likelihood that an attack would be successful but also information concerning the overall objectives of the flight as well as information on other actions that could be undertaken by the flight. For example, the primary objective of the flight may be simply to move from point A to point B, to patrol an area engaging hostile aircraft only if attacked, or to aggressively pursue any hostiles that are detected. By contrast, once a decision is made to attack, the relevant information concerns which aircraft is best able to attack a particular hostile, what actions are necessary to cover friendly aircraft, and so forth. The spectrum of information required and considerations necessary for the two decision processes thus differ substantially, although the basic information on which the decision processes are based--the detection and identification of the hostile aircraft--is common to both decision processes.

Similarly, the information required from the mental model to conduct decision processes differs from level to level. The mental model is constructed principally through the situation perception function, which, as described earlier, is action oriented, i.e., it is concerned with the calculation of those variables that are necessary to the generation and evaluation of alternative courses of actions. Since the courses of action considered at different levels differ, the information required of the mental model to carry out these processes also differ.

In summary, the different decision levels use different information (with the principal exception of the raw or interpreted sensor data), employ mutually exclusive parts of the mental model, and are concerned with the generation and evaluation of different courses of action. For this reason, the multi-tiered decision element generally employs distinct mental models and conducts the situation perception, the generation and projection of alternatives, the value association, and the alternative selection activities separately for each decision level. By contrast, the data interpretation activity, such as identifying a detected flight as hostile, are generally conducted in common for all levels. The multi-tiered decision element thus consists of a common basic information pool and a number of distinct mental models and processing activities, each of which is tailored to the decision required at a given level of the decision hierarchy.

Two methods are commonly used in value-driven decision theory for transmitting information between levels of the hierarchy.[1] The first and most obvious method is to use the decision at the upper level to restrict the lower level to examining only courses of action that are consistent with the upper level decision. This approach leads to a reduction in the number of courses of action that would otherwise have to be examined and thus serves to reduce the running time of the simulation. On the other hand, the approach does not permit the inherent flexibility and utility of the value-driven method to be fully exercised. Thus, instead of restricting the set of alternatives available for consideration, the means generally adopted is to convey the "intent" of the decision between the levels. For example, in TAC BRAWLER the decision at the flight tier for Aircraft A to attack Hostile 1 is conveyed to the pilot tier by increasing the weight assigned to those parts of the value function

[1] For distinct decision elements, the transmission may represent a direct communication between elements, e.g., between the flight leader and a wingman.

which tend to motivate an attack on Hostile 1. By "modulating" the pilot's value function in this manner, one insures that, other things being equal, Aircraft A will attack Hostile 1, while retaining the flexibility for the aircraft to adopt an alternative course of action should the situation warrant it.

The use of the value structure also plays a useful role in facilitating user selection of tactics for play and in the generation of new tactics. In most combat models, tactics are hardwired into the model. In most ground combat simulations, for example and as will be described later in the chapter, tactics such as "attack," "hold," and "delay," are hardwired into the simulation and a user is restricted to play the simulation with only these tactics. He has no flexibility for using different tactics or for generating new tactics or new tactical combinations. By contrast, the value-driven approach is naturally suited for this type of play. The user of a value-driven decision system can scale the values of alternative tactical sets, so that courses of action consistent with the tactics he wishes to play are almost always selected. He simply specifies in his input data the scaling or proportionality factors to be used in multiplying the simulation-generated values.

The capability for generating new tactics or tactical combinations rests on the capability for building "global" tactics from elemental tactical building blocks. For example, in TAC BRAWLER, even though the set of alternative courses of action at each decision point is predefined, actions are updated, i.e., new decision points are reached, every second. Thus, the resulting tactical maneuver or combined tactic although built up from predefined building blocks, is dynamically generated by the simulation. New "tactics" can thus be developed using the simulation. This proves to be a particularly valuable capability in analyzing a situation, such as the multi-aircraft combat scenario modeled in TAC BRAWLER, for which a comprehensive and fully accepted tactics set is yet to be devised.

D. COMPUTER IMPLEMENTATION OF THE VALUE-DRIVEN PARADIGM

As described in the last section, the value-driven decision system provides a basic paradigm for modeling human decision processes in computerized combat simulations. As with any computerized model, however, the development of a practical and useful simulation depends on the ability of the designer to capture the essence of the process without becoming involved in excessive or unnecessary detail. This means that the resulting simulation necessarily involves compromises in which certain parts of the decision process are omitted from the simulation or are represented by extremely simplified computer algorithms.

Experience so far in the application of the method suggests that in most simulations the detailed representation of decision elements can be limited to a few specific command functions that are central to the specific simulation. In this section the simplified decision structure used in TAC COMMANDER is described and used to illustrate how the logical structure described in the last section is implemented in an actual combat simulation.

As described in Sec. I-A, TAC COMMANDER simulates the tactical command-and-control system for NATO forces in Western Europe. Reconnaissance aircraft are committed to specific areas to gather information on the activities of enemy ground forces. This information is then communicated back to a central decision center where it is interpreted, prioritized, assigned to a decision cue, and used to assign strike aircraft to targets.

The computer implementation for TAC COMMANDER is shown in Fig. I.3. Notice first the "Simulation Arrays" at the top of the figure. These contain the current state of the "real world" as it is represented in the simulation. The exact location of all ground forces, the position, altitude, and velocities, of all aircraft are all specified in these arrays. As will be seen below, this information is available to the decision

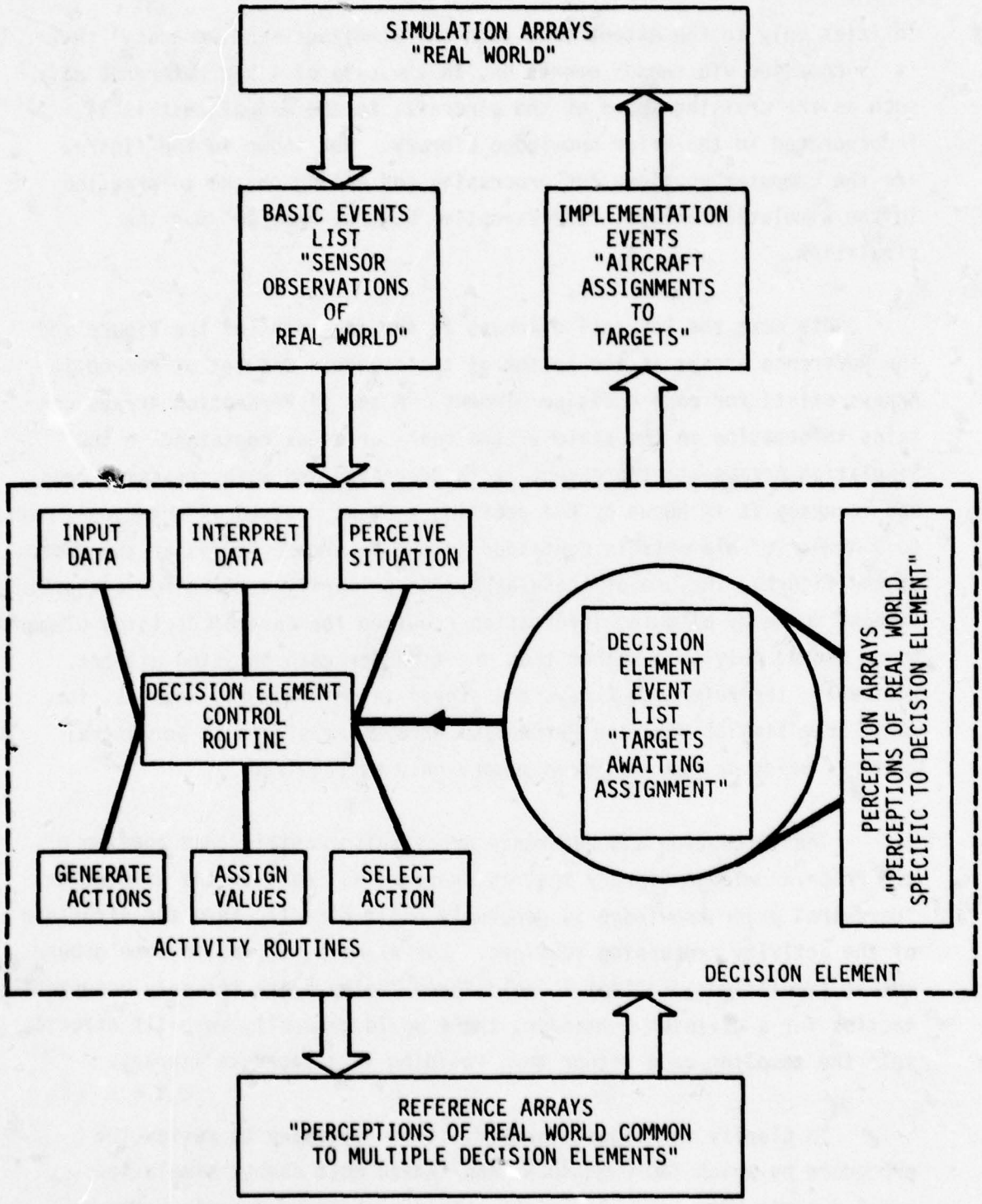


Figure I.3. Computer Implementation of TAC COMMANDER Decision Element

entities only to the extent (and with the timeliness and accuracy) that it is conveyed via sensor events or, in the case of fixed reference data such as the cruising speed of the aircraft, to the extent that it is incorporated in the Prior Knowledge Library. Not shown in the figure are the computer routines for processing and acting on the information in the simulation arrays, "The Executive Control Program" for the simulation.

Note next the Perception Arrays at the far right of the figure and the Reference arrays at the bottom of the figure. One set of Perception Arrays exists for each decision element. A set of Perception Arrays contains information on the state of the real world--as contained in the Simulation Arrays--to the extent it is perceived and with the timeliness and accuracy it is known by the decision element. Fixed information common to a number of elements is contained in the Reference Arrays at the bottom of the figure. The use of these arrays is primarily a means for conserving storage space by allowing information required for several decision elements to be stored only once rather than one time for each decision element. Typically, the Reference Arrays are stored in main memory, whereas, for very large simulations, the Perception Arrays are stored on peripheral devices, being called into main memory only as required.

The Perception and Reference Arrays also contain that portion of the Prior Knowledge Library that is provided as input to the simulation. "Hardwire" prior knowledge is generally built directly into the structure of the activity processing routines. For example, if, as in some ground combat simulations, "attack," "hold," and "delay," are the only acceptable tactics for a division commander, these would generally be built directly into the computer code rather than residing in a separate library.

To clarify the figure further, it is necessary to review the procedure by which TAC COMMANDER, and indeed most combat simulations, model dynamic behavior. In contrast to representing dynamic behavior as a continuous process, as might be appropriate in a model of a physical

fluid, TAC COMMANDER models dynamic behavior by means of dynamic events, each of which occurs at a specified time and each of which prompts the calling of a specified set of routines to perform a specific sequence of actions. Such simulations are formally known as "event-sequenced" or "event-store" simulations, because the events are stored as they are generated in a set of arrays or remote files for later processing. In TAC COMMANDER, sensor events are generated when a reconnaissance aircraft detects enemy activity--perhaps an armored column moving toward the front--and reports it back to a tactical command-and-control center.

TAC COMMANDER differs slightly from most simulations in that certain basic sensor events are pregenerated and stored for processing in simulation, i.e., "real" time. (In fact, they are pregenerated only up to the time the reconnaissance aircraft are reassigned, for the reassignment influences the detections made in the next time period.) The events are stored in the Basic Events list shown in Fig. I.3. Each event has associated with it a group of defining attributes: type of event (sensor), type of observation (armored column), location (sector 5), and, most important, the time of the observation (1410). This set of events provides the sole basis--with the exception of the information in the decision element's Prior Knowledge Library--for the decision element's perception of the real (simulated) world.

Associated with TAC COMMANDER, as with all event simulations, is an implicit clock which tracks the simulation time. It consists of a pointer that moves from event to event in the chronologically ordered Basic Event List. As an event is reached it is transferred, generally with an appropriate time delay, to the relevant decision element.

The decision element consists of the components contained within the dotted line of Fig. I.3. The Executive Control Program comprises the Decision Element Control routine and the Activities routines. The Decision Element Event List (DEEL) is properly part of the Perception arrays. Its function is described below.

An event transferred from the Basic Event List to the decision element is processed under the Decision Element Control Routine, which as a first step calls the Data Input Activity Routine. In TAC COMMANDER this routine introduces uncertainties into the location of events, e.g., the grid coordinates of the armored column; adds communication delays; and in some cases "loses" events. The Data Activity Routine may thus be best regarded as a peripheral to the decision element proper, introducing data uncertainties and data losses that model the imperfection of sensor reports and serve to "fog" the decision element's picture of the real world.

In TAC COMMANDER the Data Interpretation Activity Routine is not called for each event. The data activity function is concerned primarily with the construction of the situation map. TAC COMMANDER requires such a map only in assigning reconnaissance aircraft to sectors at the beginning of each time period. The assignment is made in accordance with the activity observed during the previous time period. Thus, the Data Interpretation Routine is called only periodically and then for the purpose of assigning reconnaissance aircraft to sectors.

The Situation Perception Routines are used in TAC COMMANDER for determining the relative importance of responding promptly to a sensor event by assigning strike aircraft to it. The determination is made in accordance with the perishability of the target, i.e., the time before the target must be relocated in order to be attacked. For example, mobile targets are highly perishable and must therefore be ascribed a high priority for attack. The output of the Perception Activity Routine is thus an indication of the importance of staging an early strike against the target.

The events are next entered into the Decision Event List, shown enclosed within the circle in Fig. I.3, which is, in effect, a decision cue containing events for which decisions must be made by assigning

aircraft to the targets represented by the events. The Decision Event List properly belongs to the Perception Arrays, for it contains that portion of the mental model corresponding to the description of the enemy activity as perceived by the decision element. Other contents of the mental model consist of the status of friendly aircraft, i.e., their home airbase, their flight status, their current assignment, and so forth.

The generation and projection of courses of action, the association of values with the actions, and the selection of an action for implementation are performed by the three remaining activity routines in Fig. I.3. A course of action in TAC COMMANDER consists of selecting a particular aircraft of a particular type from a particular airbase and assigning it to a target. Values enter into the selection and assignment process in two ways: first, since the selection and assignment of an aircraft to a target is assumed to take a finite time and the targets are assumed to be examined sequentially, it is important to make assignments for the most perishable targets first. This is treated in TAC COMMANDER by associating a time-dependent value with each perishable target that is constructed to reflect the urgency of attacking the target.

Values are also associated in TAC COMMANDER with the aircraft themselves. These values reflect the opportunity cost that is incurred if a higher value is detected after an aircraft has already been assigned to another target and the capability that is lost if the aircraft is destroyed in the attack. This is a type of Lagrange value in the sense discussed earlier. It reflects a resource constraint, its values varying dynamically according to the current availability of aircraft.

The target that is selected for attack and the aircraft that is assigned to it is that combination for which the "profit" or the difference between the target values and the aircraft value is maximum.

To implement the assignment of aircraft to targets, "implementation" events are generated by the decision element. These events, which are processed in an event list similar to the Basic Event List described above, control the assignment of the aircraft to the targets and follow the aircraft through its mission and return for reassignment.

The repetition of this procedure for each target and for new targets as they are detected and received at the decision center constitutes the essential features of the value-driven method as it is applied in TAC COMMANDER. It should be noted that the computer implementation differs in some respects from a strict application of the logical paradigm of the value-driven method. This "tailoring" of the paradigm to specific applications is a general characteristic of the method and is the subject of much of the implementation methodology described in Chapters II and III.

E. COMPARISONS WITH OTHER METHODS

In this section a perspective of the value-driven decision approach is developed that allows some of the more subtle features of the approach to be illustrated and the approach to be compared to other methods for treating command and control in combat simulations.[1] As a preliminary step in the presentation, the decision-rule approach, which is the most common procedure for treating command and control in ground combat simulations, is described and compared to the value-driven approach.

The decision-rule or the closely related decision table approach provides the simplest method for representing dynamic decision processes in combat simulations. In a representative scenario, two opposing

[1] The comparison of the method with the methods of artificial intelligence is treated as a special topic. It is addressed in Sec. F.

commanders must determine the posture, e.g., attack, hold, and delay, which their respective forces should assume during the next time period. The determination is made by comparing the ratio of some measure of the strength of each force, most commonly firepower scores, with a set of prespecified thresholds which serve as breakpoints for selecting postures for the forces. For example, a three-to-one force ratio might serve as the breakpoint for the stronger force initiating an attack and for the weaker force adopting a hold posture, whereas an eight-to-one ratio would cause the weaker force to adopt a delay posture. Typically, the estimates of force strengths are made using actual force strengths, although estimates reflecting uncertainties in the strengths are occasionally employed.

The significant features of the decision-rule approach for comparison with the value-driven method are:

- All decisions are based solely on the current state of the forces. No projections of the consequences of adopting a particular course of action are made in the simulation. Moreover, the courses of action that are considered, such as the attack, hold, and delay postures cited in the example, are "hardwired" into the software. No capability is available for dynamically generating courses of actions.
- All decisions are made by comparing selected measures, such as the firepower scores, to predefined, generally inputted threshold values.

The decision-rule approach thus lacks the richness of the value-driven approach, in not allowing for the dynamic generation, projection, and evaluation, of alternative courses of action using value criteria dynamically tuned to the state of the system. Other features of the value-driven approach, such as the data interpretation and situation perception function, while formally permitted in the decision-rule structure are generally not developed in the form or with the degree

of richness of the value-driven method. Finally, the value-driven command language for representing the flow of information among decision elements in the command-and-control system is, of course, not present in the decision-rule approach. Thus, in spite of its desirable simplicity, the decision-rule approach does not possess the versatility that is needed to adequately represent command and control in most combat simulations.

With the discussion of the decision-rule approach as background, the value-driven approach may be viewed from a much more general perspective that highlights a number of special features of the method and which further clarifies the relationship of the method to other approaches for representing the decision process in combat simulations. The development of the perspective rests on a consideration of the projection horizon, i.e., the distance into the future the courses of action are projected, and, in particular, the consequences of varying the horizon.

To make the argument explicit, consider the ground commander's decision problem as posed at the beginning of the section from the perspective of the value-driven approach. As in the decision-rule approach, the commander must select a force posture, i.e., a course of action, for the following day. (The courses of action considered may now be more general however; penetrations, envelopments and the like may be considered that depend on the current state of the war.) To assess a course of action the commander must project the action into the future. Ideally, in the sense of winning the war, it would be desirable to project the action, or more precisely the dynamic series of decisions corresponding to the action, to the end of the war. Values could then be associated with the outcomes in the form (+1, 0, -1) corresponding to win, lose, or draw, respectively, or, in a more realistic representation, to a more extensive range of values reflecting casualties incurred, length of the war, and so forth. Probabilities would be required to reflect stochastic events and uncertainties in the "state of the world." (Some allowance should also be made for the presence of a rational opponent, although in simple models his behavior is usually represented stochastically.) If all possible actions were projected in this manner and

values were then associated with the actions, the selection of the preferred action would then reduce to a problem in classical optimization theory. The value-driven method, when all alternatives are considered and when the projection horizons are extended to the end of the war, thus becomes equivalent to the classical optimization problem.

There are several drawbacks to such an approach. The number of alternatives that must be considered and the computations associated with projecting each alternative to the end of the war is nearly always computationally infeasible. In most cases, it is even difficult if not impossible to enumerate all of the possible alternatives. More fundamentally, however, such an approach, even if implemented, would not produce credible results. A commander cannot consider all alternatives nor project them to the end of the war. A sensible decision element must behave in a manner that would not exceed the cybernetic (computational) resources of the decision-maker. This generally means that only a few alternatives can be considered--generally one at a time until a satisfactory alternative is found--and that the alternatives can be projected only short times into the future.[1] The classical optimization approach thus rarely provides a satisfactory approach for representing the decision processes in combat simulations.

The second limiting case--the zero time projection--can with some qualification be viewed as a reduction of the value-driven method to the classical decision-rule method described above. More precisely, the decision-rule approach can be defined as the single alternative, zero-time projection limit of the value-driven decision method. The single alternative condition implies that values are not associated with alternatives (since there is only one) and that the determination is based solely on the current state of the system.

[1] This would not apply, of course, for a system like the Merck Scheduler, which is explicitly designed to improve and expedite the human decision process.

The comparison between the value-driven and the decision-rule approaches can be viewed from the perspective of the model designer. In the decision-rule approach, the model designer, before building the model, examines the consequences of pursuing alternative courses of action, relates them to the initial state, and then by building fixed decision rules into the model tries to ensure that the model dynamically selects desirable alternatives. In the value-driven approach, the model itself explores the consequences of pursuing alternative courses of action, associates values with their outcome, and selects a preferred action.

The more fundamental difference, however, between the value-driven method and either the decision-rule or the classical optimization approach lies in the values that must be developed in the value-driven method. In contrast to the classical optimization approach, in which the values are the "ultimate" game values (e.g., win, lose, or draw), the values in the value-driven method are associated with courses of action projected only a short distance into the future. These values therefore can only indirectly reflect the ultimate game values. Such values are called heuristic or judgmental. Their nature is most clearly illustrated in the Merck Scheduler in which the ultimate values are concerned with the maximization of profit but in which the projections provide information only on product inventories a short distance into the future. Values therefore had to be developed that would reflect the correspondence between the projected inventories and the ultimate values. As an example, a low projected inventory for a product would suggest that a stockout might occur, which would adversely impact profit. A high value should therefore be associated with putting the product into production. (Low inventories, however, mean low inventory holding costs. The association of values with production alternatives is not a simple procedure.) The illustration of procedures for generating appropriate heuristic values constitutes a major portion of this report.

F. COMPARISON WITH METHODS OF ROBOTICS

The formal structure of the value-driven decision approach as outlined in Sec. C closely resembles a similar approach used in many artificial intelligence applications. The primary difference concerns the emphasis on the higher level decision processes and the systematic use of value criteria in the value-driven approach. In most robotics applications, the robot is not designed to be self-motivating but is simply asked to execute rather well-defined activities.

The similarities between the two methods can be illustrated by considering the example referred to earlier in the discussion of the Prior Knowledge Library. For concreteness, this discussion will consider the mobile robot developed by the Stanford Research Institute and described and analyzed by Dr. Michael Arbib in his book, *The Metaphorical Brain.*[1] In the example discussed by Dr. Arbib, the mobile robot--using a television camera for viewing and wheels for locomotion--is in a room containing a large cube, two small cubes, and a sofa. A command is given to the robot to move the large cube through an open door located on one side of the room. The two small cubes are positioned so as to obscure certain of the paths the robot might select in first moving to the large cube and then "pushing" the cube from the room. The complete situation, together with two paths the robot might select in moving to the large cube, is shown in Fig. I.4.

Dr. Arbib identifies four features required for the robot to execute the command to push the large cube through the door:

1. A set of receptors which can sense the world, and a set of scene analyses which enable it to dissect sensed data in terms of interactively meaningful relations.

---

[1] Michael A. Arbib, The Metaphorical Brain, Wiley-Interscience, 1972.

2. A set of effectors, together with a set of routines well-suited to act upon the environment and to move the receptors during scene analysis.

3. An internal model which comprises an up-to-date record of the result of the system's various scene analyses and actions.

4. A problem-solver which can take reports from scene analysis to update the internal model, can compile commands into courses of action, and when necessary can interrupt other activities to update the model and replan action.

Consider these steps as they are realized in the actions of the robot and as they relate to the activities of the value-driven decision system. So that the robot can interpret the command and plan for its execution step one is concerned with the observation of the room and the perception of the objects therein. The observation of the room corresponds to the "Input Data" activity of the value-driven decision system. The errors that are simulated for that activity in the combat simulation are realized naturally in the robot through, for example, errors in range--the robot uses a rangefinder to estimate range--and "indiscernabilities" in contour lines, i.e., the edges of the cubes will generally not be clearly delineated, so that the robot cannot immediately identify a perceived object with a "description" of it stored in its data banks.

The interpretation of the incomplete series of lines as a cube corresponds to the "Interpret Data" activity of the value-driven decision system. Dr. Arbib, in fact, refers to this function of the robot as the construction of a "floor plan," a term which is a direct analog to the term "situation map" used in the definition of the Interpret Data function. Moreover, the functions performed--the identification of the cube from only partial knowledge of its characteristics and the identification of an armored column from knowledge of the presence of a few of its elements--are clearly corresponding activities.

The need for a Prior Knowledge Library similar in concept to that in the value-driven combat simulation is also a requirement for the robot. Without the prior knowledge of the characteristics of a cube, the robot would have been unable to correctly identify the object as a cube.

The second step, the set of effectors, is realized in the value-driven method by the issuance of orders by the decision elements to "effectors," which implement the decisions of the element. Such orders could include directives to the sensors to provide additional information.

The third step, the internal model, corresponds directly to the internal mental model in the value-driven approach. The mental model in both approaches provides the basis for the generation and selection of courses of action. A slight distinction between the two definitions may exist, in that the mental model for the robot emphasizes the up-to-date record of events and scene analyses, whereas in the value-driven method, the situation perception function, which is not explicitly identified by Dr. Arbib, generates parameters that explicitly enter into the generation and evaluation of alternative courses of action. The sources of this difference in emphasis probably lies, however, in the conceptually simpler problem faced by the robot which allows the perception functions to be performed either by the scene analyses routines or by the problem-solver described in step four.

The primary differences between the value-driven approach and Dr. Arbib's paradigm lies in step four, which corresponds to those activities in the value-driven approach that use the mental model in the generation and selection of courses of actions. Dr. Arbib's paradigm, while allowing for the explicit generation of courses of actions as exhibited by the two candidate routes shown in Fig. I.4, does not employ a value approach for selecting among alternatives. An explicit command, i.e., the moving of the large cube from the room in the example, is given to the robot, which then generates feasible routings first to the large

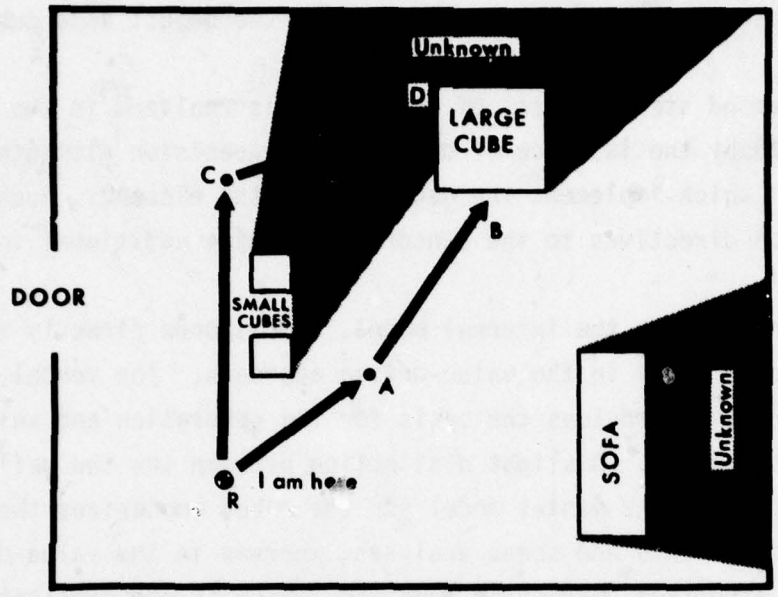


Figure I.4. Internal Model of the World as Perceived by the Robot. Black arrows identify two possible routes robot (R) could follow to reach large cube. (From M. A. Arbib, The Metaphorical Brain.)

cube and then to the door. A decision-rule-based procedure is then used for selecting an alternative for implementation. In Fig. I.4, for example, the route passing to the right of the small cubes is preferred to the route passing to the left of the cubes, since it does not pass through a region possibly containing hidden obstacles.

The corresponding paradigm for the value-driven method, as applied to a combat simulation, would also require the explicit generation of feasible courses of action, but would then associate values with the actions in accordance either with value criteria explicitly built into the simulation or dynamically computed by the simulation from more fundamental measures of merit used by the simulation in executing the decision-making function.

On the basis of step four, other comparisons can also be made between the two methods. The problem solver, together with the scene analyses routines, corresponds very closely to the Executive Control Program in the value-driven method. More interestingly, however, the hierarchical decision process described earlier for TAC BRAWLER is also used by the robot in executing the command to move the large cube through the door. The initial command is first broken into the separate commands "go to the large cube" and then "go to door, pushing cube". Next, feasible routings are examined for executing the command, and a specific route is selected. This route is then refined, so as to specify the precise trajectory the robot must follow to be in position to push the block. Finally, the detailed routing plan is translated into specific commands for turning the wheels and for advancing them the appropriate number of revolutions to move the robot along the selected route. This final step corresponds to the "subconscious" level described for TAC BRAWLER.

There are thus considerable similarities between the robot-control paradigm as described by Dr. Arbib and the control paradigm of value-driven decision theory, the fundamental differences concerning the method by which courses of action are evaluated and selected. This similarity and the emphasis in most artificial intelligence research on the basic information processing functions required for implementing the paradigm suggests that there should be opportunities for using the techniques of artificial intelligence in the Input and Interpret Data and the Perceive Situation activities of the value-driven approach. The application of these techniques will be discussed in greater detail in later chapters.

Finally, it is worth noting that although heuristic values have not been extensively used in robotics, they have frequently been used in more sophisticated problem-solving applications, such as chess and theorem proving. Thus, the value-driven approach to combat simulation can be viewed as a technique that combines the artificial intelligence methods of robotics and heuristic problem-solving to provide a representation of the human decision processes in combat simulations. Although this analogy is roughly correct, it is also somewhat misleading in that the systematic use of values as developed in the value-driven simulation has not previously been a part of the artificial intelligence tradition.

G. SUMMARY

Command and control is one of the most significant factors contributing to the performance of forces in combat. With the introduction of automated sensor systems, semi-automated command-and-control centers and C^2-linked weapon systems, the contribution of information systems to the performance of forces is becoming of even greater significance. Nevertheless, the representation of command-and-control processes in combat simulations has either been totally neglected--thereby generally assuming perfect command and control--or because of the difficulty in

representing it properly has been wholly inadequately treated. Value-driven decision theory on the other hand provides a method that has been demonstrated to have the richness and versatility that is needed to more adequately treat command and control in combat simulations. The principal advantages of the method can be conveniently summarized as follows:

- Realism of the representation. Each decision entity identified as playing a crucial role in the representation of command and control is represented in the simulation by a distinct decision element. The decision element has access only to information legitimately available to it through its sensor and communication links to the outside world (as represented by the simulation). The logical structure of the decision element is similar to the logical structure of the human decision process and parallels procedures employed in many artificial intelligence applications. This similarity makes it particularly easy to reflect in the decision element both the procedures employed by and the cybernetic limitations of the human decision-maker. Doctrinal and tactical considerations, for example, can be readily reflected in the element.

- Power of the approach. The dynamic generation of alternatives in response to changing combat conditions gives the approach the power to generate realistic courses of action either to exploit a developing situation or to prepare a suitable defense. The use and dynamic generation of heuristic values similarly gives the approach the capability to effectively evaluate the desirability of pursuing a particular course of action. In addition, the value-mediated command language permits commands to be transmitted between decision elements in the form of value preferences, thereby "encouraging" but not demanding concurrence from subordinate elements.

- Flexibility of the approach. The use of values also permits the user to easily modify tactics for play in the simulation without modification of the software. The user need only vary

the value weights associated with actions consistent with the desired tactics. New tactics can be introduced into the simulation through the addition of tactics routines but without modification to the basic software. Finally, the simulation can be used to generate new tactics by building "global" tactics from elemental tactical building blocks.

The principal disadvantage of the method is its complexity; "more straightforward" decision-rule methods would seem to offer a simpler approach for representing command and control in combat simulations. However, the difficulty in generating suitable courses of action based solely on the current state of the simulation and the lack of flexibility to dynamically generate and evaluate alternative courses of action usually makes the "practical" application of the simpler method more difficult. The basic complexity of the command-and-control problem, the variability of the environment in which it must perform, and the flexibility of the command-and-control system to respond nearly always requires the use of the more versatile and powerful method. The emphasis then falls on developing guidelines and implementation principles for using the method most simply and effectively. The remainder of the report describes general guidelines for designing a value-driven combat simulation and software procedures for implementing the method most effectively.

II. INTRODUCTION TO SIMULATION DESIGN

A. THE IMPACT OF INFORMATION QUALITY

From the foregoing discussion, it should be clear that the proper design of a value-driven combat simulation requires some predesign preparation and analysis, beyond that required for a traditional combat simulation. Conceptually, the extension of traditional simulation methods to include the processing of information is quite straightforward. In a conventional combat simulation, the status of both opposing forces is represented in status arrays (the "real world"). As the simulation proceeds, these status arrays are interrogated to determine the outcome of events, and the status arrays are updated to reflect the resulting outcomes. An information-oriented combat simulation uses this same concept except that in addition to the "real-world" status arrays, which store the physical status of forces on both sides, there are "informational" status arrays, which store the perceived state of the simulation based on the quality and quantity of information available to each side.

The information-oriented simulation therefore has the capability to deal with decisions that must be made on the basis of imperfect information. Perhaps the best example of the differences between conventional simulations and information-oriented simulations can be found in a comparison of the game of chess and poker. Chess is a game of perfect information, in the sense that all the elements of past and present situations are known to each player. The complexity of decisions in chess is primarily a result of the difficulty of considering all the relevant future alternatives, within a limited time to determine a satisfactory move. In poker, each player has incomplete information about past and present situations. Some of the player's information is certain (e.g., the cards in his hand and the betting history), but some of it is uncertain (e.g., the cards in this opponent's hand and the probabilities of drawing certain cards). Players therefore must form conclusions about the quality

of their opponent's hands by integrating a variety of different types of information and by referring to a continually changing mental model of the state of the game to make an estimation of their status in the game.

This is not to imply that poker is a more complex game than chess, for indeed it is not. The complexity of chess relative to poker stems from the differences in the numbers of move options (decision alternatives) each player has at his disposal. In poker, each player must decide between only three options: calling, raising, or folding. In chess, however, there are a large number of options based on the types of pieces left on the board, the relative positions of the pieces, and the types of moves allowed for each piece. Therefore, in poker the quality of a player is based on his ability to properly correlate and interpret his status in the game from the sparse information available to him, whereas in chess the quality of a player is based on his ability to pick an effective move from the large number of options available to him.

Combat decisions tend to combine the complex features of both poker and chess. The combat commander must base his decision on the imperfect information available to him from his sensors and communications links, and choose from a myriad of alternatives to pick the most desirable for the perceived situation. Because of the complexity of the decision process, human decision-makers, such as combat commanders, utilize many simplifications to make their decisions. Most of these same simplifications must be used in computerized decision models if they are to provide a valid representation of the human decision-maker and also keep the computational burden within practical limits. Some of the more important simplifications used by the human decision-maker are listed below, from the viewpoint of a combat pilot.

1. <u>"Outcomes" are rarely projected to an ultimate outcome</u>. A pilot does not attempt to think all the way through an air engagement. He thinks only a little ahead but tries to take actions that help him now and leave him in a "favorable position" for the remainder of the engagement.

2. Intermediate "outcomes" are evaluated in terms of judgmental criteria. Since "outcomes" are projected only a little way into the future, the pilot must use judgmental value criteria to evaluate intermediate outcomes in terms of how favorable they are likely to be for the rest of the engagement.

3. The judgmental value criteria may be adjusted to reflect current priorities. For example, the risk a pilot will be willing to accept to engage an enemy aircraft may depend on tactical priorities that have been established by his commanding officer, and possibly by his state of mind.

4. The search of alternatives is always incomplete. Unlike a mathematical optimization system, the pilot rarely, if ever, has time to consider all possible action alternatives. He considers a few that are promising and settles for one that seems more desirable (or less undesirable) than the others.

By incorporating these same simplifications into the computerized model of the combat commander, the value-driven technique simplifies the problem of simulating the decision processes that drive the simulation.

B. TIME CONTROL AND EVENTS

Among the operational issues that designers of combat simulations face is the question of the method of simulation time control. There are essentially two options for depicting the progress of the combat over time: event-sequencing or time-stepping. The underlying assumption in an event-sequenced simulation is that simulated entities (e.g., threats, sensors, decision-makers, tanks, aircraft, etc.) interact primarily through the occurrence of events that take place at discrete times. Time enters into the simulation therefore in the form of a schedule or time-line of the various events. The time for a future event is dynamically determined by the occurrences which precede it. The time-step simulation, in contrast, assumes that the combat is a continuous process that can be approximated by a series of discrete equally spaced time steps. In such a

simulation all variables of interest are modified and the state of the simulation is updated at fixed periodic intervals.

For a large-scale combat simulation, the event-sequenced simulation offers some important advantages over the time-slice approach. To understand these advantages, consider the operation of a typical event under control of an executive program. Structurally, the executive program refers to a list of future events of differing types (weapons launch, battalion movement, aircraft maneuver, etc.) and chooses the earliest event in the list for processing. Control is then transferred to a subroutine, or module, of the simulation that carries out the detailed calculations for the event. The event module interrogates the status arrays to determine the state of the simulation, performs its operations, and then modifies the state of the simulation. (In the value-driven simulation, the perceived state as well as the physical state of the simulation may be modified depending on the type of event.) After event processing, control returns to the executive program which repeats the loop.

Because of the mechanics of event processing, the event-based approach is, in most cases, more efficient than the time-slice approach and offers the following advantages to the simulation designer:

1. Event modules are independent and perform limited and clear-cut calculations. Therefore, changing the mathematics within a particular event causes no change to the processing of events and hence the mechanics of the simulation.

2. Since time does not enter directly into the event processing, other than to schedule future events, events that occur hours apart in the simulation (e.g., high-level strategic decisions) can be efficiently processed together with short-time events (e.g., aircraft maneuvers during a dogfight) in one simulation. Thus unnecessary calculations are minimized.

3. Interfaces between events are made purely through the status arrays in the simulation, thus improving the clarity of the model.

Figure II.1 shows how these two status arrays are linked in a typical value-driven simulation. Data is input into an initialization subsystem which controls the mechanics of reading user inputs, allocating storage space, and initializing appropriate values in central status. The executive program of the simulation is the event-store subsystem which provides all the processing necessary for the scheduling and execution of events in the simulation. To accomplish this, it uses the memory management subsystem to store new events into the event file as they are generated, assures that each event occurs at the proper time, and calls the appropriate execution routines to simulate the event. Results from the individual events are therefore used to update both central status and internal status (where appropriate) and produce an output file of the simulation.

Of course, the sequencing of events within the value-driven simulation, while processed by the event store subsystem, is controlled by the decision elements within the simulation. The logical relationship between a particular decision entity and its associated events is shown in Fig. II.2. Notice that the decision entity is buffered from the real world by the sensor and communication events and thus must make its decision based only on information contained in its mental model. Information can only be entered into the mental model by a selective process in the information processing event and can be utilized only by interaction between the mental model and the decision event. A particular decision entity therefore can base its decision only on that information to which it would legitimately be entitled in a real-world situation.

Proper buffering of the decision entity from the real-world status arrays requires that both the interfaces between the decision element and the central status arrays and the interfaces between individual decision elements (usually in the form of communication links) be properly treated in the simulation. Beginning with the decision entity itself, the following sections discuss some of the major considerations facing the simulation designer as he approaches the treatment of these interfaces.

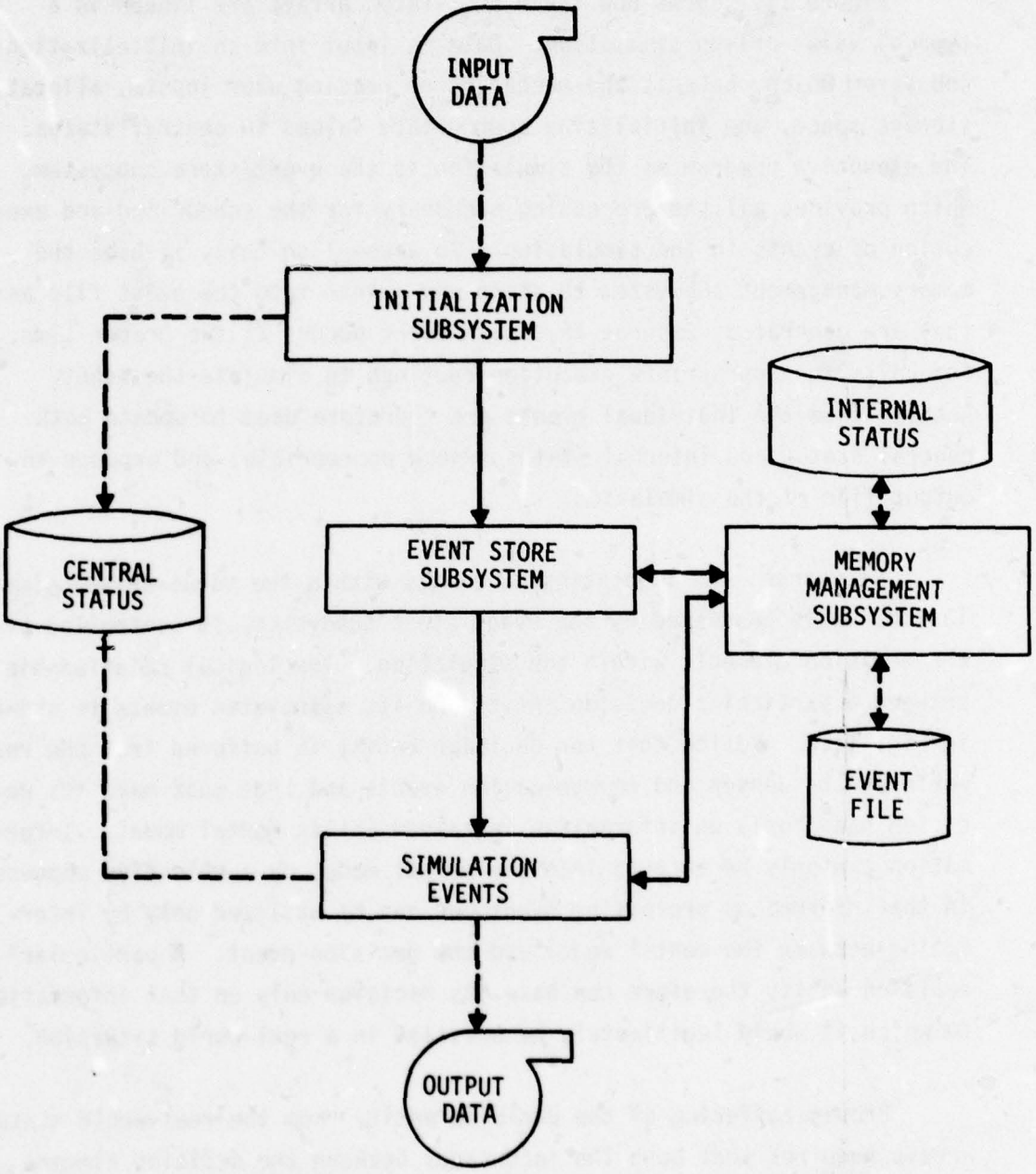


Figure II.1. Typical Simulation Structure

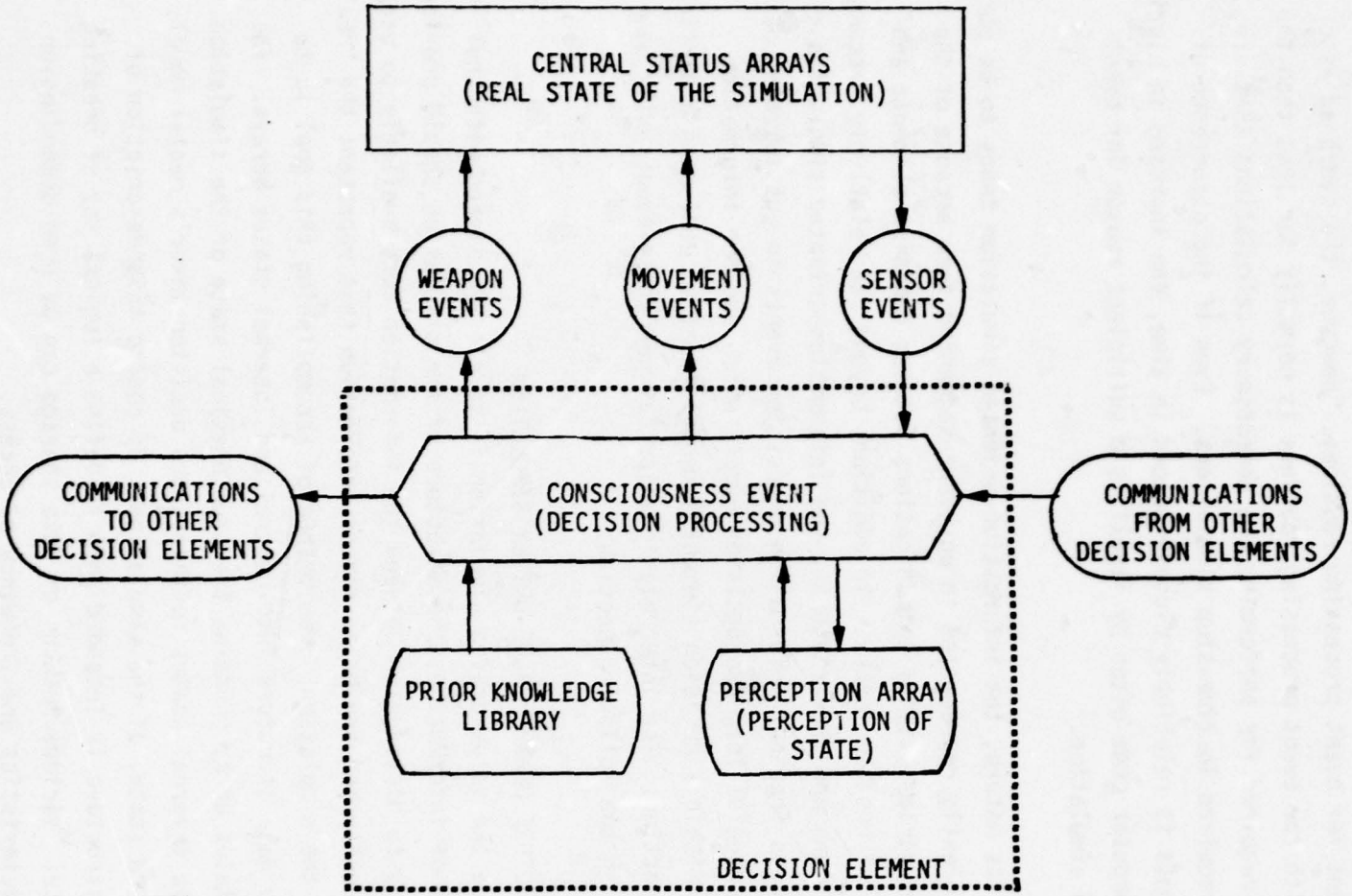


Figure II.2. Decision Entity/Event Interfaces

Of course, event processing requires some overhead due to the requirement for event processing routines. However, the overhead associated with the event processing routines is normally far less than the overhead required for performing the unnecessary calculations that are usually required in time-step simulations. Even if the occurrence of major events is relatively closely spaced in time, the increase in clarity of the computer code often by itself is a sufficient reason for event-sequenced simulation.

By its nature, the information-oriented simulation tends to be much more efficiently represented in an event-sequenced form because of the large time variability of communications between decision elements and sightings of hostile forces. In addition, beyond the relatively frequent communications and sensor events, the information-oriented simulation also requires less frequent information processing events to put incoming data into a meaningful form and decision events which use the information contained within a decision element's mental model to decide on a particular course of action. The interrelationships between these events will be discussed in the following section.

C. STRUCTURE OF THE VALUE-DRIVEN SIMULATION

Since the value-driven simulation is so closely linked with the processing of information, the structure of the simulation should provide an easy way to distinguish between the information sets available to each decision entity and the basic status information that represent the "real world" in the simulation. One method of accomplishing this goal is to divide the data structure into central and internal status arrays. The central status array contains the true physical state of the simulation, whereas the internal status contains each decision-maker's mental model, or perceived state, of the simulation. Of course this description of the data structure is intended only to define a logical way of breaking up the data. Various methods of data storage can be used depending on data characteristics and frequency of access.

D. DECISION ELEMENT DESIGN CONSIDERATIONS

The decision entity forms the nucleus of the value-driven simulation. Because of its importance, a number of questions that are critical to the proper operation and usefulness of the value-driven simulation must be addressed if the simulation is to be a realistic representation of actual combat.

Perhaps the most basic consideration facing the simulation designer is the number, structure, and relationship, of decision entities within the simulation. A large combat simulation, like actual combat, contains a great many "decision centers" ranging from the foot soldier in the field to the planning staff at the command level. Within current computational resources, it becomes clearly impractical to model each decision center from the theater commander to the foot soldier as a complete value-driven decision entity. The value-driven simulation designer must therefore perform a trade-off between the desired purpose of the simulation and the completeness with which the combat is to be modeled. If the emphasis is on the evaluation of plans and tactics, use of aggregated fighting units led by value-driven commanders should provide the required level of detail necessary to produce repeatable and accurate outcomes. However, if the emphasis of the simulation is on the interaction dynamics between low-level fighting units, use of aggregated fighting units may produce erroneous results. Ideally one would like to have a simulation that could be used to investigate both of the types of problems mentioned above. Therefore, it is desirable if simulations can be designed so that they can be restructured easily in scope, level of detail, command-and-control links, and types of units simulated, so that the same basic architecture can be adapted to a wide range of analysis problems.

The simulation of ground combat is particularly demanding in its requirements for both wide scope and extensive detail. When such a simulation is used for analytical purposes, there are sure to be cases where the desired results can be obtained by analyzing action on only a small

segment of the front, for example, a section of about Corps or Division size. Similarly there will be cases where information on overall theater outcomes can be obtained using a more aggregated representation that does not explicitly represent engagements at the battalion level. Consequently, there is a need for a very modular architecture that can be linked together much like a tinker-toy system to provide an efficient representation of specific problems of interest.

One of the key problems in providing such a variable level of detail concerns the need to have models of several different levels of aggregation. This same problem of variable levels of detail also appears in the value-driven simulation concept for an entirely different reason because of the need to provide mental models appropriate to the military commanders at the various levels. One possibility is to design the simulation in such a way that these same mental models can be substituted for the detailed simulation at some levels in order to provide a version of the simulation. This type of interchangeability in the use of the mental models will certainly not be possible unless the system interfaces are very carefully standardized to make it possible.

Ordinarily, in its normal application, a decision element will draw on information accumulated in an individual decision-maker's own consciousness array to define the status from which it will estimate outcomes for alternative courses of action. If the same model is to be used to project actual outcomes, it must, of course, calculate those outcomes based on the actual decisions made by the commanders on both sides, and it should draw on information available in the actual status arrays (rather than the consciousness array) to provide a proper starting position for the calculation. It requires careful design work to define a practical software architecture that will allow such flexibility in the use of mental models to represent different levels of detail in a combat model.

A related complication arises when a particular decision element must perform more than one basic function. Such a situation arises, for example, in the case of an aircraft flight leader. The flight leader not only must make flight-related decisions related to allocation of flight resources and assurance of successful completion of the flight's mission, but also must make lower level decisions based on his own situation as a combat pilot. Structurally, the simulated flight leader must consist of two decision processors linked to a common mental model, one processor to evaluate flight options and tactics and the other to evaluate his options and tactics as an individual pilot. In order to accomplish this, the flight leader's mental model must be expended to include additional flight-related values in addition to his pilot values. The proper treatment of value hierarchies and multiple mental models is critical to the accurate representation of the combat commander's decision process. Because this consideration is so closely linked to the value structure and processing algorithms used to simulate the decision-maker, the concept will be discussed in greater detail in Chapters III and IV.

E. COMMUNICATIONS

When new technological developments are introduced that affect the command, control, and information processes, they may also require changes in the actual command-and-control reporting structure. Thus, it is important that a C^3 model be capable of adapting easily to variations in the C^3 structure. The development of a system architecture that will accommodate such changes easily will require standardization of interfaces and a careful definition of the types of interfaces that should be interchangeable.

In order to provide such flexibility in the command-and-control system for any simulation, a careful functional analysis of the command-and-control functions must be performed to define the necessary input and output information associated with each of these C^3 functions. Once the basic functions are identified, it should be possible to design generalized modules that can carry out the same function in a variety of different command-and-control structures.

In addition, it is desirable to develop standardized message formats that are to be used within the simulation to communicate information and commands within the system. If all decision elements are designed so that they can interface to each other via this standard message format, then it is relatively simple to change the communication linkages within the simulation.

The choice of the most appropriate communication concept is not a trivial problem. A straightforward rendition of the actual message traffic would result in a totally unacceptable burden of information transmission and storage. In effect, each decision element might have to maintain his own personal copy of all relevant information received. This would lead very quickly to an information storage overload.

There are at least three approaches that can be used to minimize the problem.

1. Interpret and forget. In this approach the decision element uses each message to update the parameters of his mental model and then discards or forgets the message itself.

2. Central memory linkages. In this approach information resulting from sensor observations, etc., is stored centrally in a single list memory for each side, and the simulated "messages" really transmit only access links which make the information available to other command levels after a suitable delay. In this way it is only the access linkages that are retained by each decision entity and the basic intelligence information is recorded only once.

3. Interpret and forward. In this approach the information is interpreted to update the mental model and perhaps aggregated into a more compact and meaningful form before being transmitted to another command level.

In all probability all three options may be needed for various functions, but there is a need to define the concepts, procedures, and functions, that should be used to model the flow of information.

F. SENSORS

As mentioned earlier, the modeling of sensor interactions is one of the most time-consuming operations in most combat simulations. If the sensor interactions are processed in a routine way, it may be necessary to search all other objects within the simulation to decide what new detections may occur on a given sensor scan. If the problem of terrain line of sight is involved, it may then be necessary to make a detailed line-of-sight calculation between the two objects that are within range of each other to determine whether a detection can occur. One of the earliest ways of limiting the range of interactions in a simulation was to divide the combat area by means of a square grid divided into a number of subareas and associate each object in the simulation with one of these specific geographic subareas. In this way it was possible to limit the search to those subareas that are within range of the sensor. In this simple form, however, this technique has a number of serious disadvantages. The appropriate size of subarea to use depends on the range of the particular sensor. Moreover, no matter how the grid is chosen, a large fraction of the subareas will usually lie outside the region where forces are engaged. Finally, even when the subareas are about the right size for a particular sensor, there will always be cases when the range of the sensor will overlap several subareas so that multiple subareas must be searched.

It appears that the data processing load associated with the modeling of sensors can usually be kept within reasonable limits by exploiting a combination of techniques such as the following:

1. Use of virtual grids or geographic clustering techniques to associate neighboring objects in a way that does not require any memory space for unoccupied geographic areas.

2. Use of standardized search subroutines that can efficiently search the relevant arrays and deliver a list of pointers to all objects within the detection range (or range of interest) of a particular sensor.

3. Storage of a list of pointers to objects within potential range of each sensor so that it can be used many times before it has to be recalculated as a result of changes in the position of the sensor and its targets.

4. Precalculation of stochastic detection times so that the sensor routines are activated only for those scans when a significant event such as a detection or loss of track is predicted to occur.

5. Use of alerting messages from potential target objects to sensors when maneuvers are made that are likely to change the predicted time for detection or loss of track.

G. ADDITIONAL CONSIDERATIONS

Within each type of decision element there are also a number of important design considerations dealing with the structure and functional form of the decision element's value structure, alternative generation procedures, alternative evaluation procedures, and techniques for action implementation. Due to the relative complexity of these issues, each is discussed in greater detail in the remainder of this report.

III. VALUE DESIGN IN VALUE-DRIVEN SYSTEMS

The association of values with alternative courses of action provides the basic mechanism in the value-driven decision approach for comparing alternatives and for selecting preferred alternatives for implementation. Values are the principal feature that drives the simulation and the feature for which the greatest effort has been required in developing the value-driven method.

In this chapter, the principles and guidelines that have evolved in the development of the value methodology are described and illustrated. Following a brief examination of the classical use of values in chess as an illustration of the concept of heuristic values, the development of values as realized in the Merck Scheduler is discussed in some detail. The Scheduler provides a non-trivial but conceptually easily understood illustration of the formulation and use of heuristic values in value-driven decision systems and also provides an example of how the Generalized Lagrange Multiplier (GLM) approach has become intertwined in the development of the method. The concept of secondary or derived values is also illustrated by the Scheduler. Next the development and heuristic application of values based on the dynamic formulation of the GLM approach is described and illustrated using the TAC COMMANDER and MUSTEX models. The formulation of values in the 44 Cities study is then described to illustrate the use of multiple component value functions as a means for explicitly satisfying subsidiary constraints by selecting from among a set of nearly degenerate solutions those for which the constraints are satisfied. More advanced concepts are then described in the following chapter, where the "partitioning" of values as a means for simultaneously reflecting several objectives and the use of values as a command language for communicating between decision elements or between different levels of the same element is described. TAC BRAWLER is primarily used to illustrate these concepts.

A. THE GAME OF CHESS

Chess provides a very simple but classical illustration of the use of heuristic values as a means for guiding the short-range behavior of a decision-maker. In chess, the chess player's objectives are first to win the game and, if failing to do so, to achieve a stalemate with his opponent. In Fig. III.1 this is reflected through the assignment of ultimate or game values to the three possible outcomes--win, lose, or draw. The precise value assigned each outcome in the figure is largely arbitrary, any similarly ordered set of values differing from the specified set by a linear transformation would lead the players to the same set of decisions.

From the point of view of guiding individual moves, the ultimate game values are not very useful. Except near the end, a chess player is not able to trace the consequences of a move to the end of a game, and therefore cannot explicitly evaluate the worth of the move in terms of ultimate game values. Therefore, if he is to use values, he must devise a means for associating values with intermediate board positions, or more precisely, he must have a means for associating values with possible moves, so that he can compare the relative attractiveness of different moves, and thereby have a means for selecting a move for play.

The most common method for introducing values in chess is to associate values with the individual pieces as shown in Fig. III.1. The value of a piece, in principle, reflects its contribution toward the winning of a game, the queen, for example, being of greater value than either the bishop or the castle. The king, whose survival is imperative to the winning of the game, cannot be assigned a comparative value, for no other piece or combination of pieces is equivalent to the king. The king must survive. It is formally assigned an infinite value.

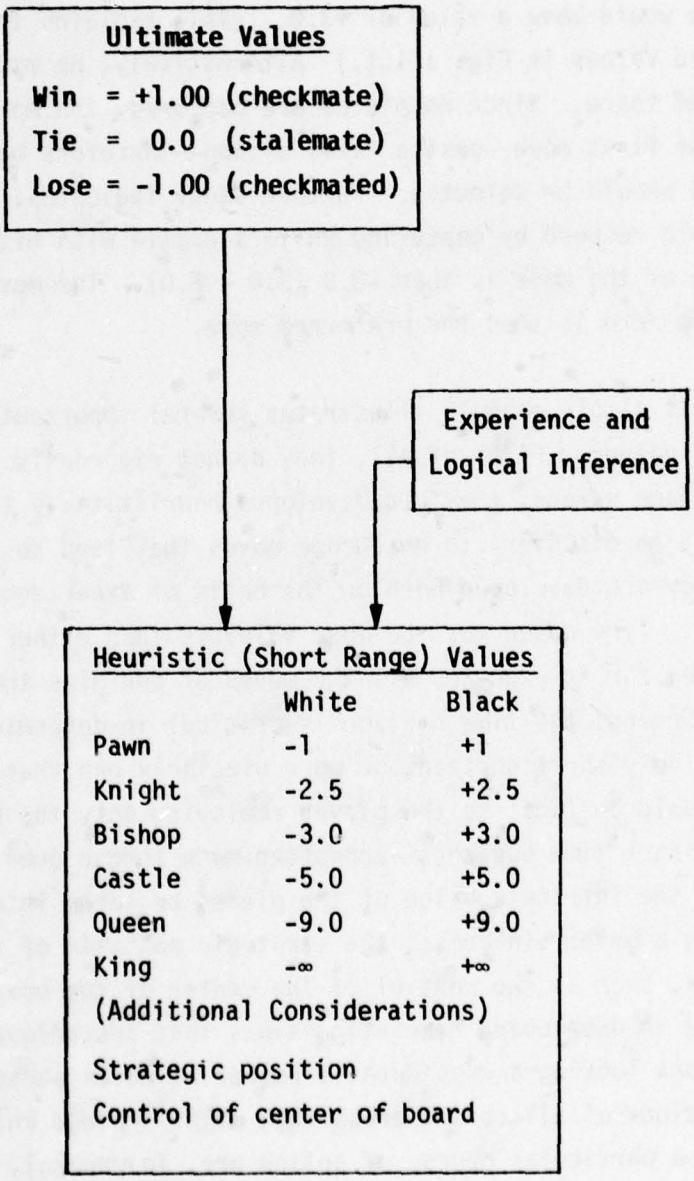


Figure III.1. Values in the Game of Chess

To appreciate the operational meaning of the values, consider how White would use the values in selecting his next move. Based on the current position of the remaining black and white pieces on the board, Black might consider capturing a black bishop with his castle. This move would have a value of +3.0. (This explains the reason for the signed values in Fig. III.1.) Alternatively, he might simply advance a pawn one space. Since no pieces are captured, the move has value zero. The first move--castle takes bishop--therefore has the highest value and should be selected. Further study indicates, however, that Black could respond by capturing White's castle with his queen. The net value of the move is then -2.0 (3.0 - 5.0). The move of the pawn with value zero is then the preferred move.

This simple example illustrates several important points about heuristic values. First of all, they do not rigorously reflect the ultimate game values; they are developed heuristically (in the sense of serving to discover) to encourage moves that tend to lead to winning play. They are developed both on the basis of experience and logical deduction. (The queen must be more valuable than either the bishop or the castle, for it executes all the moves of one plus the moves of the other.) Second, the time horizon is critical in determining the values; an excessively short horizon, or more precisely one that leaves White in danger, could be fatal to the player employing only the Fig. III.1 values. Short time horizons--and often much longer ones--require that more than the intrinsic value of the pieces be taken into account in evaluating a move. In chess, the strategic position of the pieces or its surrogates, such as the control of the center of the board, must be considered in developing heuristic values that are adequate for governing play without looking a considerable number of moves ahead. Such careful considerations of all consequences that might impinge on the desirability of taking a particular course of action are, in general, necessary in formulating values in value-driven decision systems.

Fortunately, through experience in developing value-driven systems, guidelines and principles have been identified which considerably simplify the task of developing values for value-driven systems. Although the development of such values is still largely an art, considerable guidance can be given to the designer of a value-driven system. The examples discussed in this chapter are intended to illustrate the guidelines and principles useful in the development of value structures.

B. THE MERCK SCHEDULER[1]

As indicated in the previous section, even though guidelines and general design principles for value-driven decision systems can be identified, the design of the value function is still primarily an art. It is therefore of utmost importance for the prospective designer of a value-driven system to develop an intuitive understanding of the way value functions are constructed. For this reason, we begin the discussion of value functions in value-driven systems by examining in considerable detail the development of values in the Merck Scheduler. In this system, the value structure serves as the vehicle for scheduling a large, 200 product chemical plant, which makes use of multi-purpose equipment--tanks, filters, centrifuges, and the like--that can be connected in "erector set-like" fashion to produce a production line for a particular product. Although somewhat more complex than is required for most applications, the value structure in the Scheduler provides an ideal case study for value design, in that it illustrates many of the basic principles of value function design, while avoiding such issues as uncertainty or game theoretic considerations which are important but tend to unnecessarily complicate the discussion.

1. The Scheduling Problem.

To clarify the specific scheduling problem, Fig III.2 provides a representative--but abbreviated--list of products that are required for production in a one-year time period.

---

[1] G. Lucas et al., A Modernized Plant Design and Scheduling System, in five volumes (Merck and Company), December 1970.

CARD SET TWO: TWO TEST MODULE REQUIREMENTS

PRODUCT	INDEX	PRODUCTION RATE	LIQUID	IRATE	MODULES
4 ROSALIND-III	1	79800.00	0	0.1065	2 -4 4 5 7 -7 15 15 16 19 22 24 33 37 54 -86
5 ROSALIND-II	2	33432.00	0	0.1065	-15 15 18 18 19 24 24 86
6 ROSALIND	3	61828.00	0	0.1065	300 301 303 304 305 306 312-310 -95
11 NATALIE	4	77784.00	0	0.1065	1 2 7 7 8 18 19 22 24 31 54 86
19 THEODORA	5	103992.00	0	0.1065	2 4 4 -5 8 -8 16 98
20 SHARON III-A	6	28056.00	0	0.1065	7 7 18 -18 19 19 24 24 -24 31 54 86
21 SHARON II-A	7	43680.00	0	0.1065	1 5 15 15 16 -24 -24 26 63
22 SHARON IV-B2	8	61588.00	1	0.1065	17 18 18 -19 -19 -24 -24 24 24 31 34 47
23 SHARON III-B2	9	179560.00	1	0.1065	1 2 5 -90 -24 24 24 26 31
24 SHARON III-B1	10	69048.00	1	0.1065	90 90 18 19 19 22 24 26 -37 -37
25 SHARON IIB	11	243600.00	1	0.1065	1 2 5 18 -19 -19 -24 24 24 31 37
26 SHARON	12	87192.00	0	0.1065	1 1 -2 -2 -2 -8 16 24 37 37
32 MELISSA II	13	105840.00	0	0.1065	1 2 4 5 9 90 15 15 16 18 18 19 19 -91 -23 -24 -24 24 24 24 29 31 33 -37 -37 -37 37 37 -37 -11 47 47 54 54 66 86
33 MELISSA	14	91560.00	0	0.1065	-305-306-306-306-312 312 312 310 310 310 310 310 95 95
.	.	.	.		
.	.	.	.		
.	.	.	.		
.	.	.	.		
.	.	.	.		
.	.	.	.		
.	.	.	.		
.	.	.	.		
.	.	.	.		
.	.	.	.		
51 MARY	27	162456.00	0	0.1065	302 304 307 309 95 95 317 317
52 SARA-LEA	28	58800.00	0	0.1065	301 304 306 308 -95-317-318
53 SARA	29	55440.00	0	0.1065	-303-306-313-310-310 316

Figure III.2. Representative Product and Module--Centrifuges, Filters, Tanks, etc.,--List for the Merck Scheduler.

Girls' names are used to identify the products, so as to protect Merck proprietary data. The types of products involved, however, include Vitamin C, Monosodium Glutamate (MSG), and Indomuthicin. The most significant information in the figure is the list of modules for each product. These modules are the filters, centrifuges, etc. that must be connected together to produce a specific product. Note that multiple modules of a given type, such as the two module 4's for Rosalind-III, can be required to produce a given product. A minus sign indicates that a substitute module may be substituted for a primary one. A separate list of substitute modules is separately input to the Scheduler.

A highly simplified production schedule is shown in Fig. III.3. The products, numbered 11-108, run across the top of the figure with the time periods, each of one week's duration, running down the left-most column. The "x's" in the body of the figure indicate that a product was produced during a time period; the "."s" indicate that a product was not produced during the period. In the figure note that typical production runs for the products are from two to five weeks. These relatively short run times, which reflect trade-offs between setup times and inventory holding costs, result in products being put into production from one to five times per year and are hence, responsible for the complexity of the schedules in the Merck plant. Finally, note that the rather sparce scheduling of products indicated in the figure--not more than five products are in production at any one time--is a property of the limited product set used for illustration. As will be shown below, the sparcity of products in production does not persist when a full product set is used.

The modular usage for the schedule of Fig. III.3 is shown in Fig. III.4. The modules are shown across the top--there are three modules of type 1, for example--and the periods are shown down the far left-hand column, as before. The numbers within the matrix are the identities of the products using the specific modules. For example,

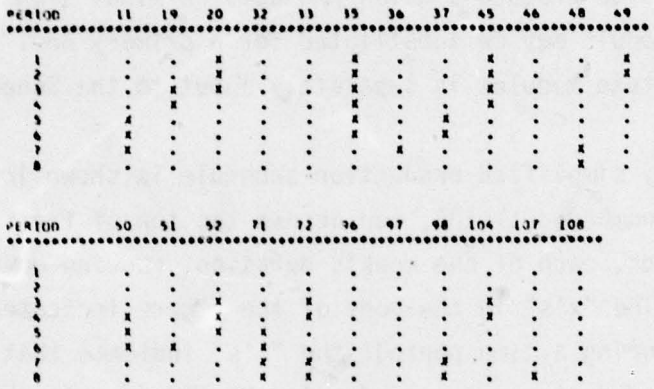


Figure III.3. A Sample Schedule for a Small Product Set.

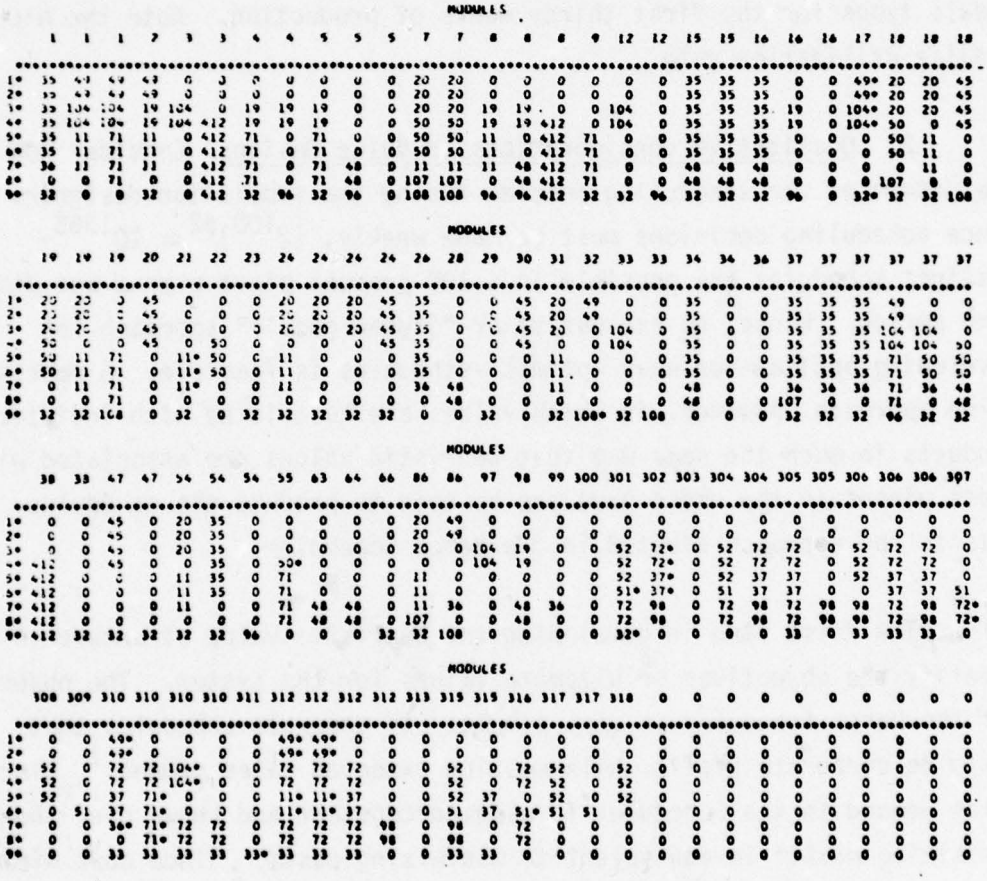


Figure III.4. Modular Utilization For The Schedule Shown in Fig. III.3.

product 35 uses the first of the modules of type 1 during the first six time periods. An asterisk beside a product--e.g., product 11 in module 21 in periods 5-7--indicates that a substitute module is used. Another product has already been assigned the primary module.

To indicate the nature of the schedules for a full product set, Fig. III.5 indicates the schedule for such a set for the first three module types for the first thirty weeks of production. Note the high modular utilization rate.

2. Qualitative Considerations in Value Design. Consider now the nature of the scheduling problem facing the simulation designer. Since scheduling decisions must be made weekly, $(2^{100})^{52} \simeq 10^{1565}$ distinct schedules are possible in a 100-product plant over a one-year time period. Hence, no exhaustive or "jigsaw puzzle" approach for developing optimal--or near-optimal--schedules is feasible. A heuristic value approach, however, in which values are associated with individual products in much the same way that heuristic values are associated with chess pieces in the chess game can be used to produce the schedules. This is the approach adopted in the Merck Scheduler.

The first step in developing the heuristic value structure is to identify the objectives or ultimate values for the system. The objective for the Merck Scheduler is very simple: to generate schedules that maximize corporate profit while meeting expected sales demand.[1] Since sales demand in the Scheduler is assumed constant and known over the year, maximizing profit is equivalent to minimizing costs. Since cost minimization provides a conceptually simpler way to think about the scheduling problem, it will be used in the following discussion.

[1] Strictly speaking, in developing schedules maximizing profit takes precedence over meeting sales demand. For a properly sized plant, however, stockouts occur infrequently, since large penalties reflecting loss of goodwill are "charged" against the schedule.

	1	1	1	1	1	1	2	2	2	2	2	2	2	2	5	5	5	5	5	5	5	5
10	19	49	49	131	105	0	19	49	52	77	131	131	0	0	0	19	19	49	52	52	59	59	77	117
20	26	49	49	106	106	0	26	49	52	91	91	106	106	0	0	91	106	49	52	52	59	59	0	117
30	26	54	115	106	106	115	26	54	52	91	91	106	106	115	115	91	106	54	52	52	59	59	115	117
40	132	0	115	106	106	115	132	132	52	91	91	106	106	115	115	91	106	60	52	52	60	132	115	117
50	5	5	71	71	0	0	5	5	52	91	91	71	0	0	0	91	5	60	52	52	60	188	188	117
60	75	75	71	71	116	116	75	90	90	91	91	71	116	116	0	91	75	60	90	116	60	188	188	116
70	31	101	71	71	104	0	52	90	90	101	104	71	0	0	0	31	31	52	90	52	104	188	188	104
80	31	101	32	35	105	0	32	90	90	101	32	35	0	0	0	31	31	35	90	35	59	59	72	72
90	36	0	32	35	105	0	32	90	90	36	32	35	0	0	0	136	136	35	90	35	59	59	188	188
100	36	104	106	106	137	0	104	90	90	36	106	106	137	137	0	136	136	104	90	104	106	137	188	188
110	38	104	75	75	105	0	104	90	90	38	75	77	0	0	0	37	37	104	90	104	75	77	188	188
120	38	33	46	106	106	137	46	90	90	38	106	106	137	137	0	33	46	46	90	74	74	106	137	137
130	119	119	133	106	106	33	119	119	133	133	106	106	0	0	0	72	72	88	88	119	119	106	133	133
140	119	119	32	75	75	78	119	119	32	32	75	77	90	90	0	37	37	75	77	119	119	78	90	103
150	35	106	32	106	121	0	35	106	32	32	106	0	90	90	0	35	35	60	60	106	121	121	90	103
160	35	36	105	0	121	0	35	36	0	0	0	0	90	90	0	35	35	87	87	188	121	121	90	188
170	28	28	48	48	106	106	28	28	48	48	106	106	90	90	0	28	28	48	48	188	74	74	90	188
180	33	36	46	105	0	0	36	46	0	0	0	0	90	90	0	33	46	46	72	188	72	88	90	188
190	31	96	46	106	106	0	96	46	106	106	0	0	90	90	0	31	46	46	31	96	96	106	90	0
200	29	32	75	106	106	75	32	32	106	106	75	77	90	90	0	29	75	77	0	0	0	106	90	0
210	2	32	19	78	102	183	32	32	2	49	52	102	90	90	183	19	19	52	52	59	59	78	90	183
220	114	116	116	134	0	0	77	114	114	114	116	116	90	90	134	74	74	77	114	114	116	116	90	134
230	71	71	97	106	106	0	52	71	97	97	106	106	90	90	0	37	37	52	52	97	106	0	90	0
240	48	48	75	106	106	75	48	48	75	0	106	106	90	90	0	48	48	72	72	75	106	88	90	88
250	33	104	0	0	0	0	104	0	0	0	0	0	90	90	0	22	33	44	44	104	104	0	90	0
260	31	71	71	106	106	0	71	106	106	0	0	0	90	90	0	31	31	106	0	0	0	0	90	0
270	31	71	71	24	32	14	71	24	32	32	77	0	90	90	0	31	31	14	34	39	77	0	90	0
280	75	75	97	106	106	0	75	97	97	106	77	106	90	90	0	75	97	106	136	136	77	0	90	0
290	105	0	0	0	0	0	0	0	0	0	0	0	90	90	0	72	72	88	136	136	88	0	90	0
300	5	5	33	46	78	0	5	5	46	0	0	0	90	90	0	5	33	46	136	136	46	78	90	0

Figure III.5. A Representative Portion of a Schedule For a Full Product Set.

The next step in developing the value structure is to identify the features that a schedule should possess in order to minimize costs. These may be conveniently described first for an oversized plant in which the products may be scheduled independently and then for the normal or undersized plant in which the products compete for the use of the modules. The value structure must then be constructed so that the schedules will exhibit the desired features.

In the oversized plant, cycle lengths, i.e., the time between the start of consecutive production runs for each product, are the primary feature used to characterize the schedules. They are constructed so as to minimize the cost of production for the products. The determination of an optimal cycle length for a product involves a trade-off between the cost of setting up the modules for production and cleaning them out afterwards and the inventory holding costs, which reflect the cost of storing the product. Setup and cleanout costs vary inversely with the cycle time, for if the cycle time is halved, the number of setups required is doubled, and the inventory holding costs vary proportionally with the cycle time. Formally, then, the cost per unit of product produced, excluding the raw material costs, is given by:

$$C = \frac{\alpha}{T} + \beta T$$

where α and β are readily calculated constants. The optimal cycle time is then given by:

$$T_{opt} = \sqrt{\frac{\alpha}{\beta}}$$

which shows that cycle length increases with setup costs and decreases with inventory holding costs. The product run lengths implicit in Fig. III.3 reflect the trade-off between these two factors.

Having determined the optimal run length, it is necessary to determine the conditions under which a product is put into production. These conditions should be related to the inventory of the product on hand. For example, a product with a low inventory on hand and in danger of a stockout should be put in production before a product with a large inventory on hand. In fact, it should be possible, based on economic considerations, to define a normal minimum inventory level, at which point the product would go into production. The product would then continue in production for a run time corresponding to the optimal run length of the product. This run time would then define a normal maximum inventory level. Thus, in the oversized plant the inventory level should fluctuate between normal minimum and maximum inventory levels with production beginning when the inventory on hand drops to the normal minimum level. If a product were prematurely to go into production at an intermediate inventory level, it would incur excessive setup costs if it cut off at the normal maximum inventory level, or, if production continued for a normal run time, it would incur excessive inventory holding costs. The heuristic value function must therefore implicitly look ahead to foresee the added cost burden that would be incurred if the product is put into production at a point other than the normal minimum inventory level.

Now consider the situation that occurs in the normal or undersized plant. Competition for use of the modules now arises among the products. A determination must then be made of which products are more important to produce during the current time period and which can be postponed to subsequent periods. Postponement of production causes the normal oscillations between minimum and maximum normal inventory levels to be distorted. "Bottleneck" modules--which are modules required by many products or, more explicitly, modules with high utilization rates--arise which stand to dominate the development of a schedule. Products using these modules must, in effect, be scheduled first and other modules then scheduled around them. The scheduling of the normal

or undersized plant, which is by far of greater interest as modules are immensely expensive, is thus much more complex, involving the trade-off of many more factors, than is the scheduling of the oversized plant.

To reflect the competition between products and to ensure the most profitable schedule is produced, the most useful procedure to use in constructing a value function for a product is in terms of the "cost" one is willing to incur to produce on the product. The higher values, i.e., a greater willingness to incur costs, should therefore be assigned to very profitable products rather than to the less profitable products. Equally important, however, in developing satisfactory schedules is that products using bottleneck modules should have higher values than products not using as many bottleneck modules, for they are harder to schedule and should therefore be given a higher value (i.e., one should be willing to incur a greater cost to produce them). Somehow included in the value function should be a reflection of the "value" of the modules required for the product's production. These "values" will be shown below to correspond to Lagrange Multipliers, or in economic language, shadow values. They represent (approximately) the added profit that would be realized by adding one additional module of the specified type to the plant.

3. Mathematical Preliminaries: General Lagrange Multiplier Theory. Before describing the construction of the values in detail, it is useful to assume they are already at our disposal and to indicate how they may be used to select the optimal product set for a given time period. The discussion will also illustrate the nature and role of the GLM in the value-driven theory. The reader may, if he so chooses, skip much of the mathematics to follow. He should, however, understand the physical interpretation of the Lagrange Multiplier or shadow value. It will appear again in the discussion of values for the TAC COMMANDER and MUSTEX models.

As a first step, the scheduling problem is cast in the form of an integer linear programming problem. In this form, the set of products to be produced is that set which maximizes the total product value, subject to the constraints on the availability of modules:

Maximize

$$Z = v_1x_1 + v_2x_2 + \ldots + v_Nx_N$$

subject to the constraints

$$\begin{aligned} n_{11}x_1 + n_{12}x_2 + \ldots n_{1N}x_N &\leq N_1 \\ \cdot \qquad\qquad\qquad\qquad &\quad \cdot \\ \cdot \qquad\qquad\qquad\qquad &\quad \cdot \\ \cdot \qquad\qquad\qquad\qquad &\quad \cdot \\ n_{M1}x_1 + n_{M2}x_2 + \ldots n_{MN}x_N &\leq N_M \end{aligned}$$

where

v_i = the value of product i.
x_i = {0,1} is the production variable for product i.
n_{ij} = number of modules of type i used by product j.
N = number of products.
N_i = number of modules of type i.
M = number of module types.

In this form, the problem resembles a standard linear programming problem except that the variables assume integer rather than continuous values. Since the variables may only assume the values (0,1) continuous approximations to the variables are not feasible.

The GLM theory may now be applied to the problem. This will not only allow the problem to be solved but also will provide the shadow values that are required in the construction of the value functions themselves. A formal statement of the GLM theorem for discrete variables is as follows:

Given: A set of N + 1 functions $Z(X)$, $C^K(X)$, $K = 1, \ldots, N$ defined on a set S, $X \varepsilon S$.

Then: If λ^K, $K = 1, \ldots, N$ are sets of nonnegative real numbers and X^* maximizes the function

$$L(X) = Z(X) - \sum_{k=1}^{N} \lambda^K C^K(X)$$

over the set S, then X^* maximizes $Z(X)$ over the set of all X such that $C^K(X) \leq C^K(X^*)$ for all K.

Note that the theorem very closely resembles the corresponding Lagrange theorem of Advanced Calculus. That theorem, however, requires the functions $Z(X)$ and $C^K(X)$ to be continuously differentiable over the set S. By contrast, in this theorem the set S may be any set, including the nonnegative integers, and the functions $Z(X)$ and $C^K(X)$ may be any functions, continuous or otherwise, having only the property that the maximum of $L(X)$, known as the Lagrangian, exists.

It is important to realize that a price is paid for the generality of the Lagrange theorem; it would not be feasible to apply without a computer. To use the theorem, one first chooses a set of Lagrange Multipliers (shadow values) λ_K and then maximizes the Lagrangian $L(X)$ by whatever means are convenient. (Note this is an unconstrained maximization.) The value X^* that maximizes $L(X)$, then also maximizes $Z(X)$ over the set of all X, for which $C^K(X) \leq C^K(X^*)$. One thus does not know for what set of constraints the problem has been solved until after $L(X)$

has been maximized [(since one does not know X^*, one does not know $C^K(X^*)$]. In solving a problem for a given set of constraints it is necessary to maximized and remaximize L(X) for various sets of λ^K's until a set which satisfies the desired constraints is found.

Numerous iterative techniques have been devised for determining the λ_K's and discussions of them are contained in the references. Almost all of them are based, however, on the physical interpretation of the λ_K's, an interpretation we will use repeatedly in this section. Restricting ourselves for simplicity to continuously differentiable functions, it may be readily shown that

$$\lambda_K = \frac{\partial Z(X^*)}{\partial X_K^*}$$

What this expression means is that λ_K represents the increase in value that would be realized by adding one additional unit of resource K. Expressed in terms of our problem, λ_K will be seen to represent the additional profit that is realized if a module of type K is added to the plant. If for an optimal solution, a constraint is not active, i.e., some of the modules are unused, then the corresponding λ_K is zero, for adding an additional module will not improve the solution.

To obtain the iterative solution, one may proceed directly from the physical interpretation or, more simply, notice the way in which λ_K enters the expression for L(X). If an intermediate solution is found that uses more $C_K(X)$ than is actually available, increasing λ_K increases the penalty for using the resource, which on the next iteration will reduce the quantity of the resource consumed. Similarly, if a resource is underused, decreasing λ_K on the next iteration will increase the quantity of the resource consumed. Proceeding systematically in this manner will generally lead to the desired solution.

The proof of the GLM theorem is very simple. It may be proved in two steps:

Since X^* maximizes $L(X)$,

$$L(X^*) = Z(X^*) - \sum_{K=1}^{N} \lambda_K C^K(X^*) \geq L(X) = Z(X) - \sum_{K=1}^{N} \lambda_K C^K(X) \text{ for all } X,$$

so that

$$Z(X^*) \geq Z(X) + \sum_{K=1}^{N} \lambda_K [C^K(X^*) - C^K(X)]$$

and therefore for all X such that $C^K(X) \leq C^K(X^*)$, $K = 1, 2, \ldots, N$

$$Z(X^*) \geq Z(X).$$

The final step follows as a consequence of the requirement that the λ_K's be nonnegative real numbers.

The final mathematical step required to complete the formalism is to put the integer linear program for the Scheduler into the Lagrangian formulation. If we make the identification

$$Z(X) = \sum_{i=1}^{N} V_i X_i \qquad X = (X_1, X_2, \ldots, X_N)$$

$$C^K(X) = \sum_{i=1}^{N} n_{Ki} X_i \qquad K = 1, 2, \ldots, M$$

then

$$L(X) = \sum_{i=1}^{N} V_i X_i - \sum_{K=1}^{M} \lambda_K \sum_{i=1}^{N} X_i = \sum_{i=1}^{N} L_i(X_i)$$

where

$$L_i(X_i) = \left(V_i - \sum_{k=1}^{N} \lambda_K n_{Ki}\right) X_i$$

= (Value of product i minus sum of shadow values for product i)X_i.

The important feature of this expression is that the total Lagrangian for the whole product set has separated into a sum of individual Lagrangians, $L_i(X_i)$, one Lagrangian corresponding to each product. The total Lagrangian L(X) may then be maximized by maximizing each $L_i(X_i)$ separately. The introduction of the Lagrange Multipliers has thus served to decouple the products. Furthermore, each $L_i(X_i)$ is easily maximized, for it assumes only two values: a zero value if the product is not in production, and a non-zero value if it is in production. If the in-production value is positive, then the Lagrangian is maximized by putting the product in production. If the in-production value is negative, then the Lagrangian is maximized by not putting it into production.

To obtain the optimal product set for a period, subject to the constraints on the modules, an initial set of λ's is chosen and the individual Lagrangians $L_i(X_i)$ are maximized. The initial λ's are set to a weighted average of the "instantaneous" λ's for the preceding periods. After the maximization is completed, the number of each type of module required is determined and compared to the number available. The λ's are then adjusted according to the procedure described above and the maximization repeated. Proceeding in this manner, the optimal schedule is generated.

4. Construction of the Value Function. Consider now the detailed construction of the value functions. Figure III.6 illustrates a representative value function for a final product, i.e., a product which can be purchased from an external source for resale. Note first that the values are a function of the inventory on hand. For large inventories, it is not critical to produce a product during the next time period, so that the values are small; conversely, for small inventories, it is more important to produce the product, so that the values are correspondingly high.

Next note the presence of the maximum and minimum normal inventories levels. As previously described, these represent the inventory levels between which the inventory would normally be expected to fluctuate. The difference between the two levels is equal to the inventory produced during an optimal run length diminished by the sales during the run. In a typical run in a plant with excess capacity, a production run for a product will normally be initiated when the inventory falls to the normal minimum inventory level. To explain how the value function ensures the inventory will fluctuate between the two levels, and how the value function modifies the behavior in a tightly constrained plant, it is necessary to examine the construction of the value function in some detail.

The value function for a product is constructed so that its value at any inventory level represents the cost per pound that one is willing to incur to produce the product. Therefore, the maximum value of the value function must not exceed the cost at which the product can be purchased from an external source for resale. This cost sets the upper limit for the value function. It is indicated by the upper dotted line in Fig. III.6.

The direct variable costs--raw material, utilities, and labor--must be charged against the product. Their cost per unit of product produced is indicated by the lower dotted line in Fig. III.6. Since it

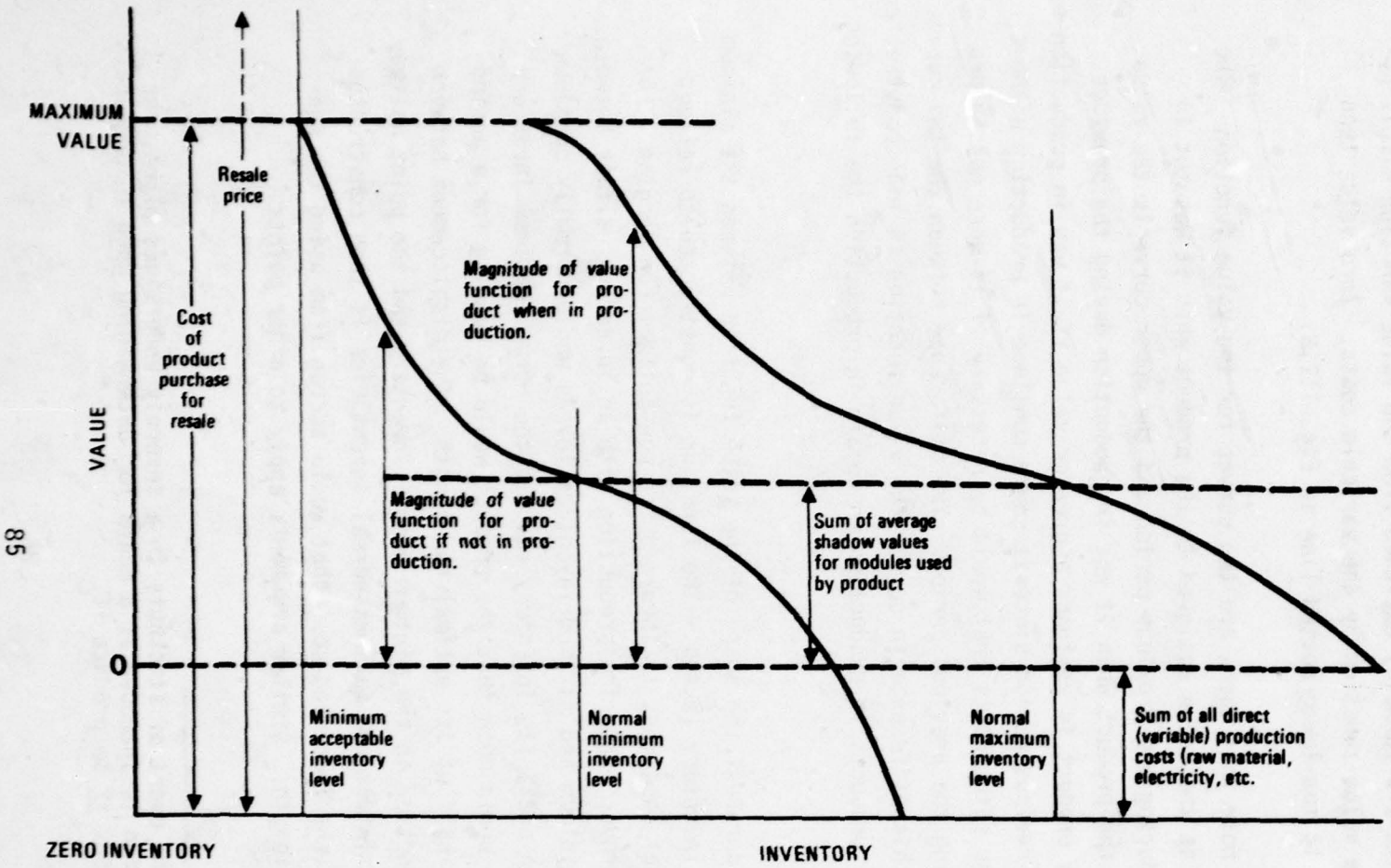


Figure III.6. Value Function for a Final Product with a Make/Buy Option

is assumed that the products will be produced during the year, the cost of the variable components can be excluded from the value function.[1] Variable costs are formally excluded from the value function simply by reducing the value function by the variable costs. Zero value then corresponds to the lower dotted line in Fig. III.6.

Next note that there are two curves for the value function. The lower curve is the value assigned to the product when it was not in production during the previous period and the upper curve is the value assigned to the product when it was in production during the previous period. The product is assigned a greater value if it was in production during the previous period since it could continue in production without incurring the setup costs that would be necessary if it were not in production during the previous period. The difference between the two curves represents this difference in cost. The value function is thus constructed to encourage products in production to remain in production the following period.

Consider next the shape of the value function between the minimum and maximum inventory levels. The function is constructed to reflect the additional cost that is incurred for non-optimal run lengths. For example, a product entering production with an inventory halfway between the normal minimum and maximum inventory levels would normally continue in production until its inventory reached the normal maximum inventory level. This would mean that the product would be running for a period equal to one-half of its optimal run length. The displacement between the value function at the minimum normal inventory and the point halfway between the minimum and maximum normal inventories is then constructed to represent the increased cost that would accrue from using the non-optimal run length. Similar arguments apply to other points.

---

[1] Except in the event of stockouts in a severely undersized plant, the value function is intended as a means for determining <u>when</u> to produce a product, not <u>if</u> to produce it.

Now consider the values of the value function at the minimum and maximum normal inventory levels. The generation of these values is the most critical factor in the generation of the value function and one of the most important features to be illustrated by the development of value functions in the Scheduler. To establish the value, first observe that there is an opportunity cost associated with the use of a module by a product. By using a module, a product denies its use to another product which could go into production were it available. There is thus a cost associated with using the module that should properly be charged to the product and hence, should be included in the value function. A reasonable measure of the magnitude of the cost that should be charged to the product is the added profit that would be realized if an additional module of the specified type were added to the plant. But this value is just the shadow value (Lagrange Multiplier) λ_K. Hence, the Lagrange Multiplier provides the quantitative measure of cost that is required in the value function.

To generate the value of the value function at the minimum and maximum normal inventory levels, it is then, with one provision, only necessary to set the value function at the two points equal to the sum of the shadow values for the modules used by the product. The provision is that the sum should be taken over the "average" shadow values, where the averages are generated by averaging the "instantaneous" shadow values over several time periods. The average shadow values are more representative of the average cost of the modules and, as will be seen below, their use ensures the proper development of a schedule in a constrained plant.

To see that, on the average, the correspondence between the normal minimum and maximum inventory levels and the shadow values results in the product going into production at the normal minimum inventory level and out of production at the normal maximum inventory level, recall that the Lagrangian for a product i has the form:

$$L_i(X_i) = \left(V_i - \sum_{k=1}^{M} \lambda_k n_{ki} \right) X_i$$

The Lagrangian may be either positive or negative. If it is positive, the product is put into production; if it is negative, it is not put into production. Therefore, if the inventory is initially at a point between the normal minimum and maximum inventory levels and the product was not in production during the preceding period, the Lagrangian is negative and the product is not put in production during the following period. As the inventory decreases, the point is reached at which the value v_i becomes equal to the sum of the shadow values and the product goes into production. This is just the normal minimum inventory level. Similarly, as new inventory is produced and the value function drops to a position where it intersects the average shadow value curve, the product goes out of production. This is the maximum normal inventory point. The value function thus behaves in the desired manner.

Before describing the behavior of the value function in a "bottle-necked" plant, it is necessary to establish the position of the normal minimum inventory level relative to the origin. For a final product value function in which the product can be purchased from the outside, it is possible to specify a minimum acceptable inventory. This inventory is essentially a buffer inventory below which the actual inventory should not drop. The position of the minimum normal inventory line is then simply chosen so that the value function intersects the minimum acceptable inventory line with a value equal to the cost of the final product. (The shape of the value function curve to the left of the minimum normal inventory level is constructed to reflect the additional cost incurred by the expected increased run time, just as the value function curve to the right of the level represents the additional cost incurred by the decreased run time.) This completes the definition of the value function.

Consider now the behavior of the value function in a bottlenecked plant, i.e., a plant in which certain modules tend to constrain the development of the schedule by being in urgent demand by two or more products. To explore this situation, it is first necessary to note the direct correspondence that exists between the bottleneck modules and the magnitude of the shadow values associated with them. Since the bottleneck modules constrain the development of a schedule, they therefore constrain the capability of the plant to generate a profit. If additional bottleneck modules are added to the plant, the modules cease to be bottlenecks and the "absolute" profitability at the plant can be expected to significantly increase. This means that the shadow value for the bottleneck modules, which represent the increased profitability of the plant resulting from the addition of a module of the specified type, are large. Hence, large shadow values correspond to bottleneck modules.

Now consider the behavior of the value function first when a product does not use bottleneck modules and second when it uses at least one bottleneck module. When a product uses no bottleneck modules, the sum of its average shadow values is small so that the lower two dotted lines in Fig. III.6 tend to coalesce. The product will then go into production at the normal minimum inventory level and will go out of production at the normal maximum inventory level, for the "out of production" value function is negative at all intermediate points and the "in production" value function is positive at intermediate points and negative beyond the normal maximum inventory level. If a product uses no bottleneck modules, it is thus produced just as it would be produced in an oversized plant.

When a product uses bottleneck modules, the space between the lower dotted line in Fig. III.6 tends to increase. This has two related effects: (1) the value functions for products using bottleneck modules becomes large, thereby encouraging their production and, (2) since the value functions are now positive in regions where they were formerly negative, production may begin earlier; i.e., for inventories less than the normal minimum inventory; and may continue longer; i.e.,

for inventories in excess of the normal maximum inventory. The increase in shadow values provides the flexibility in production scheduling necessary to construct a satisfactory schedule.

The detailed determination of the time a product using bottleneck modules is put into and taken out of production involves the interplay between the average and the instantaneous shadow values. Recalling that the value function v_i includes the sum of the average shadow values in its construction, and from the individual Lagrangian,

$$L_i(x_i) = \left(v_i - \sum_{k=1}^{n} \lambda_K n_{Ki}\right) x_i$$

that the instantaneous shadow values are subtracted from the value function in the Lagrangian, it can be seen that the difference between the two types of shadow values is critical to the determination of the sign of the Lagrangian (and hence, to whether or not the product is put into production). It is just this interplay that is the key to the generation of efficient schedules. If, for example, a few modules, which are generally bottleneck modules, happen to be available during a period--e.g., because another product that has used them now has adequate inventory on hand--then the instantaneous shadow value for them will be low. The difference between the value of a product that uses them and the sum of the instantaneous shadow values will then be large and the product will be virtually certain to be put into production--when "the getting's good," so to speak. This mechanism together with the previously described mechanism--values are high when inventories are low--which is basically independent of the shadow values, provides the central drivers for the Scheduler.

Other features of the two driving mechanisms can be readily inferred by the reader. One particularly interesting feature of the instantaneous Lambda mechanism is that it may lead not only to a product

being put into production early but to production being delayed if modules have an unusually large instantaneous demand. Thus, the instantaneous Lambda mechanism can either push the production of products forward or backward in time to develop efficient and cost-effective schedules.

5. Derived Values. Before summarizing the general features of the value functions in the Scheduler and relating them to combat modeling, it is useful to consider briefly how the concept of a derived or inputed value is realized in the Scheduler. In the Scheduler, final products are not in general produced directly from raw materials delivered to the plant. Instead, a multistage process is required in which chains of "intermediate" products are produced to provide the "raw material" necessary to produce the final product. The intermediate products are similar to final products in every way except that instead of being sold, their output is used as input to the next higher product in the chain. The products thus have derived or inputed value in the sense that they are not sold as final products but they must, nevertheless, be produced in order to produce the final product. The problem is then how to associate values with the intermediates.

The method that was found to work most effectively for reflecting the value of intermediate products was to use for each intermediate its final product value function with two modifications.

- In place of using the inventory of the final product, the intermediate product used the difference of: (1) the cumulative inventory of all products at and above the intermediate product in the chain and, (2) the cumulative sales demand projected ahead to the time when the current output of the intermediate could be expected to be converted into the final product.
- The shadow values for the modules used by the intermediate replaced the shadow values of the final product in calculating the value function.

The first modification is intended to reflect two characteristics of the production process in the value functions: (1) all inventory at or above the intermediate product in the production chain may be used to produce the final product without requiring additional production of the intermediate and, (2) the relevant time for assessing the final product inventory is the time that the intermediate inventory, which would be produced during the next time intervals, is available for use in producing the final product. The second modification simply reflects the fact that the module rent that must be paid to put the intermediate in production are rents on the modules used by the intermediate not by the final products.

The interesting feature of the derived values in the Scheduler is that the values represent an initial formative step in the development of techniques for transmitting valuative information in value-driven decision theory. Current practice primarily makes use of the value-mediated command language, which makes the transmission of valuative parameters between decision elements, for conveying information. Chapters IV and VII discuss the use of the command language in value-driven systems.

6. Implications for Combat Modeling. Now let us summarize a number of features of the value function in the Scheduler that have particularly significant implications for combat modeling.

- The most fundamental consideration in the design of value-driven systems is to develop a value structure that produces the behavior desired of the simulation. The value structure has no intrinsic value in itself but is only a means of producing the desired behavior. Even in the Scheduler, in which profit maximization seems to be the guiding principle, it is in reality but a guideline for constructing desirable schedules. Where its use results in features of the value structure that lead to undesirable features in the schedule, the value functions--profit maximization aside--are modified

to produce the desired behavior. Such was the case, for example, in determining the normal minimum inventory level from the normal maximum inventory level. Strict application of the profit maximization principle produced inventories that were judged to be too high. An ad hoc modification was then made to the value function to produce the behavior desired. Similar considerations, of course, apply in developing value structures for combat simulations. It is extremely desirable to have guiding principles analogous to the profit maximization profit for designing the value structure, e.g., maximization of territorial gain, but the fundamental consideration is that the simulation behaves in a sensible manner.

- In the Scheduler the shadow values are used directly in the construction of the value itself. Although this may seem to be a special feature designed specifically for the Scheduler, it is in fact representative of a very general approach that has been used extensively in the development of value-driven decision systems. The shadow values reflect the worth of a constrained resource--in this case, the modules used in the production of the products--and thus may be thought of as derived or inputed values in much the same way as the heuristic values derive from the ultimate values or the intermediate product values derive from the final product values. In combat applications, as is shown in the next section, the shadow values may directly assume the role of the heuristic values, serving to guide the selection of courses of action in combat. Combat aircraft, for example, are in reality resources in the same sense as the modules in the Scheduler are resources, for they are used in achieving a combat goal. They thus have shadow values associated with them and these may be used as decision-making values. For example, there is a value associated with destroying

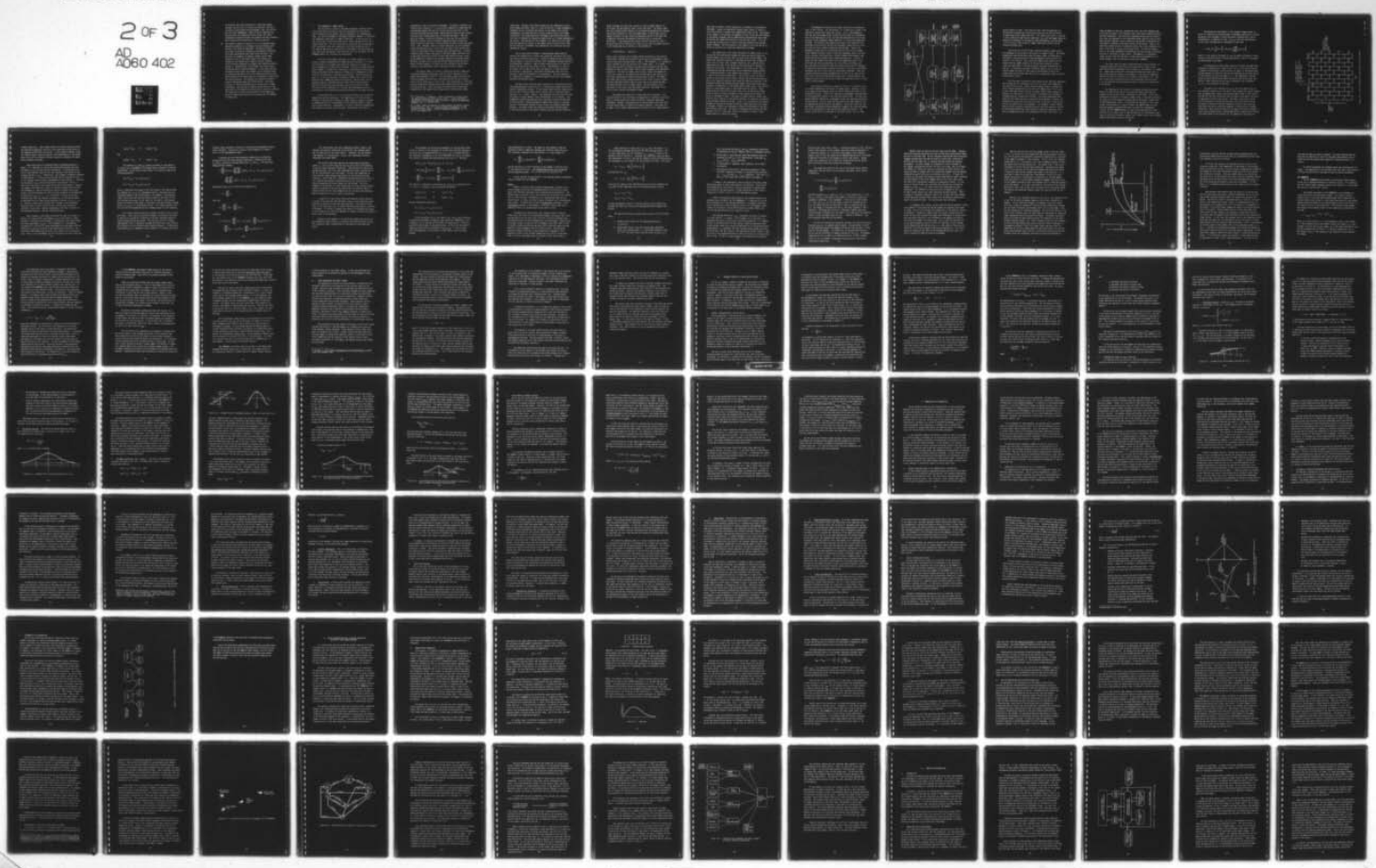
an aircraft and a value at risk in committing an aircraft to attack. The use of shadow values in the Scheduler in constructing the heuristic or decision-making values is thus a very general procedure that can be exploited in many combat applications.

- The projection of alternatives for the purpose of assigning values and selecting an alternative for implementation is nearly always conducted at a higher level of aggregating than is the actual run of the simulation. In projecting the alternatives in the Scheduler, for example, the assumption is implicitly made that the product will be able to run until the normal maximum inventory is achieved. The Scheduler in making the projection does not look at the whole product set, determine which product will be using which modules during which time periods, and then decide if the product can run for its natural run length. Rather the assessment is made at a much higher level of aggregation, using the shadow values to reflect in an approximate way the future availability of the modules. The general principle of aggregated projection is maintained in nearly if not all value-driven decision systems. It is especially important in combat applications where not only is it computationally infeasible to make the detailed projections, but it often requires access to information the decision-maker would not be expected to have available to him.

- In the Scheduler, once a product is put into production its value is increased--by an amount equal to the setup and cleanout costs--over its value when the product is not in production. This has the effect of ensuring that, once the decision is made to put the product in production, the product will--barring a major bottleneck--be produced for the full production run. This general type of value-enhancement feature is also appropriate in combat modeling.

AD-A060 402 DECISION-SCIENCE APPLICATIONS INC ARLINGTON VA F/G 12/2
VALUE-DRIVEN DECISION THEORY: APPLICATION TO COMBAT SIMULATIONS--ETC(U)
JUL 78 G L LUCAS, G F GORMAN, G E PUGH F49620-77-C-0089
UNCLASSIFIED DSA-67 AFOSR-TR-78-1398 NL

2 OF 3
AD
A060 402



In designing the value structure for a decision element (perhaps the decision to initiate an offensive), one wants to ensure, barring major new developments, that the decision will be fully implemented. Short of not re-examining the decision until the completion of the action, the value-enhancement procedure provides the simplest and most effective way to ensure that the decision is fully carried out.

- The Scheduler generally operates in a fully automated mode, generating schedules in accordance with the features built directly into the value functions. Frequently, however, one wishes to construct schedules with properties differing from those generated by the normal value functions. One may wish, for example, to produce schedules with longer production runs so that fewer setups and cleanouts are required or one might wish to impose a partial schedule and to allow the Scheduler to schedule the remaining products around it. Modifications of the former type can be accomplished by varying "tuning parameters" in the value function. Increased running times, for example, can be obtained either by increasing the setup costs or by decreasing holding costs. Modifications of the latter type are introduced by "overruling" the value functions. The products are entered as though in certain time periods they had infinite value. Such capabilities for manual manipulation are also desired in combat simulations. One may wish to employ certain tactics or may even wish to fully specify the activities of one side, allowing the simulation to select the activities of the opposition. Specific combat applications are discussed under TAC BRAWLER in Chapter VII.

C. THE HIERARCHY OF COMBAT VALUES

The discussion of values in the Merck Scheduler provides an overview of the nature of values as they are structured in value-driven decision theory and the general rules and guidelines for implementing them. The value development in the Merck Scheduler, however, is obviously not immediately applicable to combat simulations. In this section we shall explicitly consider the hierarchy of values appropriate for use in combat simulations. We shall show now the concept of the Lagrange Multiplier developed in the last subsection is of prime importance both to the rigorous and heuristic application of the theory. We shall begin by first describing the general theory and then will use the MUSTEX and TAC COMMANDER models to illustrate its application.

Since one of the major objectives in a combat simulation is to provide a better representation of human decision processes, it is generally desirable to use common-sense decision processes as a guide to the development of a formal value structure. Typically, a human decision-maker segments a problem into a hierarchy of long-range and short-range planning problems and then adopts an informal set of judgmental decision critieria at each level in the decision hierarchy. Other things being equal, it is usually desirable to develop a formal value structure that parallels this common-sense decision hierarchy. Not only does such an approach produce a simulated decision process that more accurately parallels the human decision processes, but also it results in more efficient decision processes for it allows the simulated decisions to capitalize on the accumulated wisdom and experience of the human planners.

On the other hand, it is not sufficient to simply try to parallel common-sense decision processes. The common-sense or intuitive understanding of the decision critieria is almost always too vague to provide an adequate guide for the development of the required formal or quantitative value structure. Consequently, in order to develop an appropriate and logically consistent value structure, the designer must draw on

theoretical as well as practical knowledge. To provide a logically consistent value structure, it is particularly important to understand the relationship between the heuristic values used at each level in the decision hierarchy and the longer range objectives that are more closely related to the ultimate objectives of the decision-maker. When the decision process is carefully analyzed in this way, one can usually identify a logical chain of valuative deductions which relates the value criteria at each level in the hierarchy to those at the next higher level. Thus, through a chain of logical deductions one can ultimately relate the heuristic value criteria for the lowest level tactical decisions to the highest level strategic goals and objectives. To understand the value criteria that apply in military combat, we would like to know how the heuristic value criteria that guide decisions in day-to-day combat are related to the broader objective of winning the war. Moreover, we would like to know how the war winning objectives that govern national behavior in time of war are related to the economic, political, and ideological values that apply in time of peace.

The general theory of values as applied to military combat was originally developed by George E. Pugh and John P. Mayberry in two papers that appeared in 1973.[1,2] The first paper deals with the relationship between peacetime goals and objectives and the war winning value criteria that apply in time of war. The second paper is concerned with the relationship between day-to-day combat values and the objective of winning the war. These two papers provide the basic theoretical foundation for the formal value systems used in value-driven combat

[1] G. E. Pugh and J. P. Mayberry, "Theory of Measures of Effectiveness for General Purpose Military Forces: Part I - A Zero Sum Payoff Appropriate for Evaluating Combat Strategies," Journal of Operations Research, Vol. 21, 867-885 (1973).

[2] G. E. Pugh, "Theory of Measures of Effectiveness for General Purpose Military Forces: Part II - Lagrange Dynamic Programming in Time Sequential Combat Games," Journal of Operations Research, Vol. 21, No. 4, July-August 1973.

simulations. Because of the great variability and complexity of real-world combat decisions, there is a limit on how far a formal mathematical framework can go in defining the details of the combat value system. Consequently, the detailed value criteria actually used in the combat simulations include many heuristic refinements and elaborations which (at least at present) are beyond the capabilities of a formal mathematical derivation. The objective of this chapter is to familiarize the reader with the theoretical aspects of the combat value structure and to show how the heuristic refinements and elaborations are related to the basic theoretical results.

One of the central problems in understanding combat behavior concerns the sudden shift in values and objectives that takes place with the onset of military hostilities. During the prewar period, the nations seem to behave in a way that reflects their independent economic political and ideological interests. Some of these interests may be in conflict between the two nations but others are not. Nevertheless, as soon as war breaks out the two nations begin to behave as if their objectives are directly in opposition. Each nation deliberately seeks to block the other from the achievement of its objectives. The first paper by Dr. Pugh and Mr. Mayberry is concerned primarily with this shift between peacetime and wartime values.

The paper deals with the formal structure of a two-person non-zero sum bargaining game in which there is a relationship between threats and probable negotiated outcomes. If the players choose to play cooperatively they can each achieve a payoff that is better than if they play uncooperatively (i.e., fight). However, the negotiated solution that each side can expect to obtain depends on the effectiveness of its threats. This mathematical bargaining game which was originally analyzed by John Nash in 1953 provides a simple mathematical model which clearly illustrates the basic relationship between military forces and probable negotiated outcomes. The theory shows that probable negotiated outcomes depend on the effectiveness of the threat strategy, and that the most effective

threat strategy for both sides involves a form of combat behavior in which each side deliberately seeks to hurt the other. The theory shows that this optimum threat behavior can be generated mathematically through the min-max solution of a zero sum combat game, in which each side attempts to protect its own values and objectives while at the same time attempting to prevent the opponent from realizing his values and objectives. If we represent the overall utility function for side A by U_A and the utility function for side B as U_B, then the formal solution of the combat game can be defined as the solution to

$$\min/\max\{U_A(x,y) - \gamma U_B(x,y)\}$$

where x and y represent the strategy choices available to side A and B, respectively, and the utility payoffs U_A and U_B are represented as functions of the joint strategy choice x and y. The parameter γ is a weighting factor selected to put the utilities on a common scale. It is worth noting that the utility functions U_A and U_B are concerned with the real elements of value on both sides, such as lives, property, and ideological objectives. The tools of combat, such as guns, tanks, and combat aircraft, are a means to the objective of winning the war but they are not explicitly represented in the war winning value criteria. The theory also shows that for most combat situations this value criteria dictates that the combatants should fight in such a way as to achieve the most favorable possible ratio of combat losses. For more detail on the derivation of this basic zero sum combat objective, the reader is referred to the original Pugh-Mayberry paper.

One theoretical objection is often made to this analysis. The analysis defines a zero sum threat strategy which is optimum in the sense that it should provide each side with the most favorable negotiated settlement. However, the question can be raised whether this same value criteria should apply if hostilities actually develop. Two points can be made. First, actual combat behavior (at least to first approximation)

does seem to reflect a direct opposition of interests as predicted by the model. Second, on theoretical grounds actual combat is usually a last resort which is invoked after negotiations have failed. The purpose of the combat is to convince the opponent of the reality and effectiveness of the threat strategy, both by demonstrating determination and by demonstrating the combat effectiveness of the military forces. This can be done most effectively by actually following through on the optimum threat strategy. The real purpose of the combat is to bring the war as quickly as possible to a favorable negotiated settlement.

The second step in the formal derivation of combat values is concerned with the relationship between the above war winning objective and the combat value criteria that are used in day-to-day combat. The theoretical objective function is concerned only with real elements of value, such as lives, land, economic resources, etc., whereas actual combat decisions tend to be focused on the distinction or capture of enemy military forces. The second paper in the series by Dr. Pugh is concerned with the way the military forces themselves become involved as a dominating factor in the combat value function. Intuitively, it is easy to see how this comes about. The military forces are the resources or tools that must be used to achieve the war winning objective. Just as the production modules which were the limiting resources in the production scheduling problem acquired shadow values because they were essential to production, so the combat forces (tanks, guns, and aircraft) acquire a large shadow value in the combat decision process. These shadow values on the military forces play a role in day-to-day combat decisions which is exactly analogous to that played by the values of pieces in the game of chess. Each type of combat resource (tanks, guns, aircraft, etc.) acquires a value which reflects its potential contribution to the ultimate outcome of the war. Since it is not practical to project outcomes all the way to the end of the war, combat decisions (like moves in the game of chess) are evaluated in terms of an estimate of the probable losses on each side. The tactic or strategy that promises the most favorable ratio or losses when losses are measured in terms of these shadow values will usually be chosen.

The mathematical derivation of these results will be addressed later in this section. For the present, it is sufficient to make a few qualitative observations about the results. First, the mathematics confirms that the shadow values of the military forces are exceedingly high at the beginning of a war, and that the concentration of military decisions on the opponents combat resources is in fact the appropriate strategy. The value of the combat forces tends to be highest at the beginning of the conflict and declines gradually toward the end of the war after the basic combat outcome has been decided. Second, the mathematics shows that in a protracted conflict the ratio of shadow values between different pieces of military equipment tends to be relatively constant (reflecting their relative combat effectiveness) even though the absolute shadow values change substantially. Finally, it shows that the shadow values enter into the combat decisions in two different ways that are analogous to a rent and to a purchase cost. When considering alternatives that may involve destruction of equipment on both sides, the planner considers the ratio of the value of equipment destroyed, and asks if the ratio is sufficiently favorable. When considering alternatives where equipment will be tied up in some activity but is not likely to be destroyed, he must still consider whether the probable value of the activity will exceed the opportunity cost or rental value of the equipment.

At the present time this is about as far as it is possible to go in a formal mathematical derivation of combat values. However, in practical combat decision-making there are several additional levels in the hierarchy of values. Figure III.7 provides a diagrammatic representation of the hierarchical levels in the value derivation chain. The top level, labeled National Objectives, corresponds to the real peacetime objectives of both nations. At this level the two nations have some interests that are opposed and others that are not opposed. The value criteria at the next level, labeled war winning objectives, however, are directly opposed and can be expressed as a zero sum min-max objective function. The values at this level are analogous to the zero sum win, lose, or draw

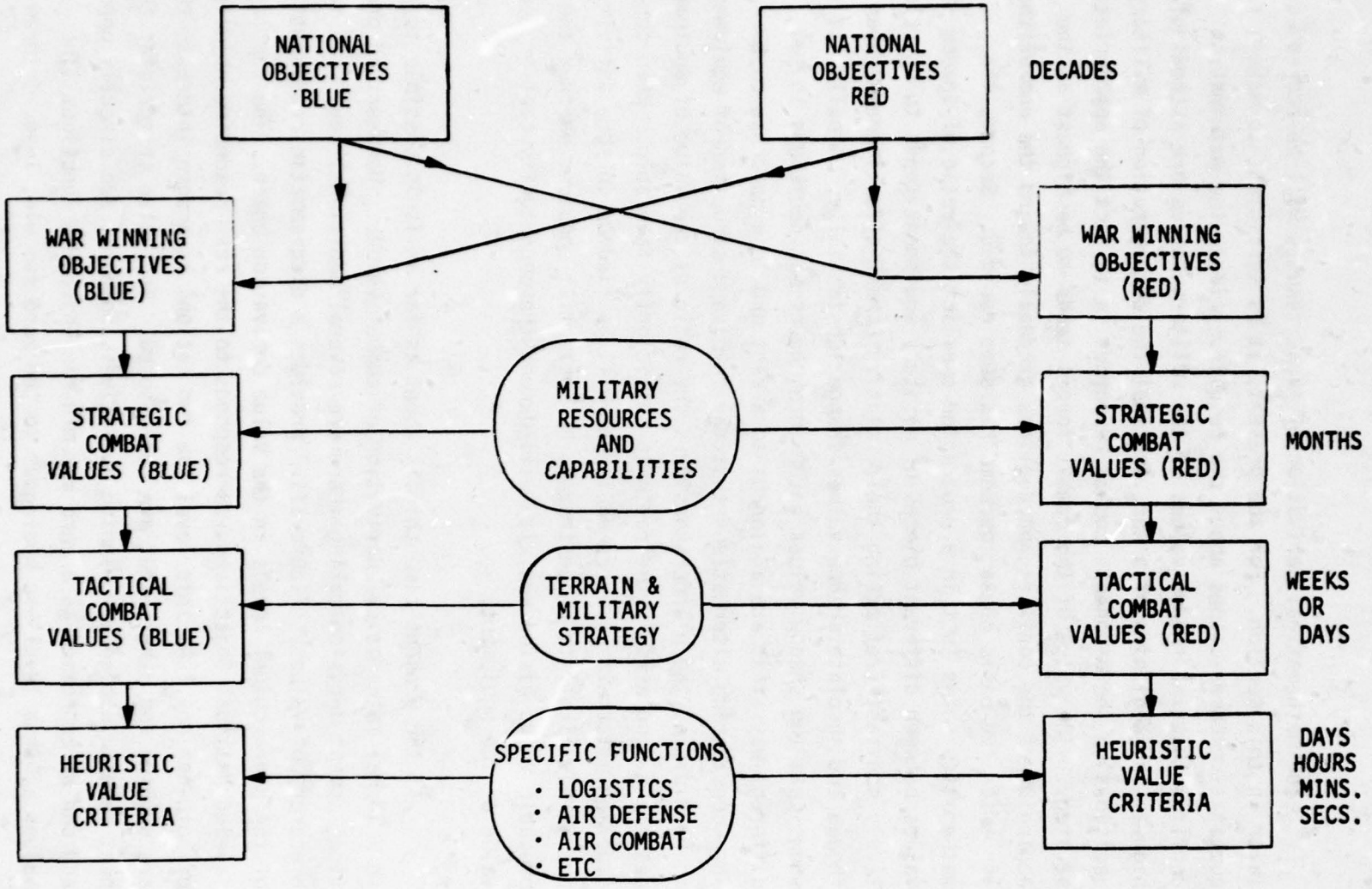


Figure III.7. Combat Value Derivation Chain

value criteria in chess. The values at this level are applicable in defining the long-term war winning objectives but are not very useful in making day-to-day combat decisions. The formal logic that is involved in this conversion from national objectives to a zero sum combat objective function is developed in the paper by Pugh and Mayberry, but the logic is not critical to the design of combat simulations and will not be discussed further here.

The values at the third level, labeled strategic combat values, are analogous to the values of the pieces in a game of chess. The formal mathematical derivation which shows how these values are generated is developed in the second paper of the Pugh-Mayberry series and is discussed in this section. These strategic combat values (which are concerned with the combat potential of the military resources on both sides) are applicable in considering major strategic alternatives for the war. In selecting such major strategic options, such as where to land on enemy territory, or what type of encirclement to plan, the planner must also consider how the specific strategy will affect expected long-term loss ratios.

Once the basic strategy has been chosen by the top-level commanders, the strategy must be implemented tactically by lower level commanders. The fourth level values, labeled tactical combat values, represent the value criteria appropriate to this tactical command level. The values at this level must differ from the strategic combat values because the objective at this level is not to select a strategy but to implement a strategy that has already been selected. To provide appropriate motivation at this level the value system must be elaborated to reflect the strategic value of controlling specific areas of terrain. These values may be, and usually are, time dependent. For example, if the strategic plan calls for a specific penetration route, then the control of terrain on the planned penetration route becomes a high value objective, which must be weighed along with the consideration of local loss ratios. In

order to adhere to the basic penetration plan, the local commander may be willing to accept a less favorable loss ratio than he would ordinarily consider acceptable. The tactically elaborated values that are used at this level are analogous to the elaborated values that are used in chess when tactical features, such as control of the center of the board, are considered in addition to the basic values of the pieces. There is no doubt that these tactically elaborated values play a very important role both in chess and in actual military decisions. At present, however, there is no formal mathematical methodology that can be used to determine the form or structure of these values. Consequently, in developing values at this level the designer is almost entirely dependent on informal methods, such as experience and military judgment.

The final level of values, labeled heuristic value criteria, are concerned specifically with heuristic values that are used in very specialized decision problems, such as logistics, air defense, air combat, etc. These value criteria are concerned with detailed issues such as the value of specific energy in air combat, or the value of engaging a hostile aircraft as a function of its depth of penetration, etc. They thus involve quick-response tactical and operational decisions where it is not feasible to think ahead more than a few hours or perhaps even a few seconds.

The remainder of this section is designed to develop the basic theory of combat values and to illustrate the theory with specific examples at the various levels illustrated in the figure. First, the theoretical derivation of strategic combat values is discussed following the logic for a time sequential combat game as developed in paper two of the Pugh-Mayberry series. This theoretical approach is then illustrated in the context of a specific model, the MUSTEX model in which the time sequential optimization logic is explicitly implemented. Finally, the TAC COMMANDER model is used to show how heuristic value criteria at the lowest level in the hierarchy can be formulated so that they are explicitly related to higher level values, such as the strategic combat values within the same simulation.

The theoretical development of the strategic combat values is based on a Generalized Lagrange Dynamic Programming approach for solving two-person zero-sum games in which the payoff function H(x,y) can be represented as a summation of terms $h_i(x_i,y_i)$ over the time periods i. In this case, the game may be represented in the form:

$$v = \max_{\underline{x}} \min_{\underline{y}} \left\{ H(x,y) \right\} = \max_{\underline{x}} \min_{\underline{y}} \left\{ \sum_{i=1}^{n} h_i(x_i,y_i) \right\}$$

where v is the value of the game, (x_i,y_i) are mixed strategies for Blue (Player 1) and Red (Player 2) during time period i; and n is the number of time periods in the game.

A representative ground combat scenario for which the game representation is appropriate is shown in Fig. III.8. This figure provides a schematic illustration of the evolution of combat in the battle areas and the sequence of decisions that must be made during each time period. In the scenario, the decision-makers, i.e., the commanders, have the option of reallocating forces among the battle areas during each time period. These reallocation decisions correspond to the strategies (x_i,y_i) in the statement of the game.

Included in the individual $h_i(x_i,y_i)$ are such combat losses as men, economic resources, and territory. Not included are combat elements, such as combat tanks, ships, and aircraft, whose value is not intrinsic but derives from the contribution they can make to the payoff and which is measured in units of real value. The tanks, ships, and aircraft are thus treated as constrained resources, their availability being expressed in ordinary constraint equations and their value deriving from their ability to assist the commander in achieving his ultimate objectives. Their explicit value will be shown to be given by dynamically varying shadow values (Lagrange Multipliers), whose value reflects the future worth of the resources to the commander in achieving his

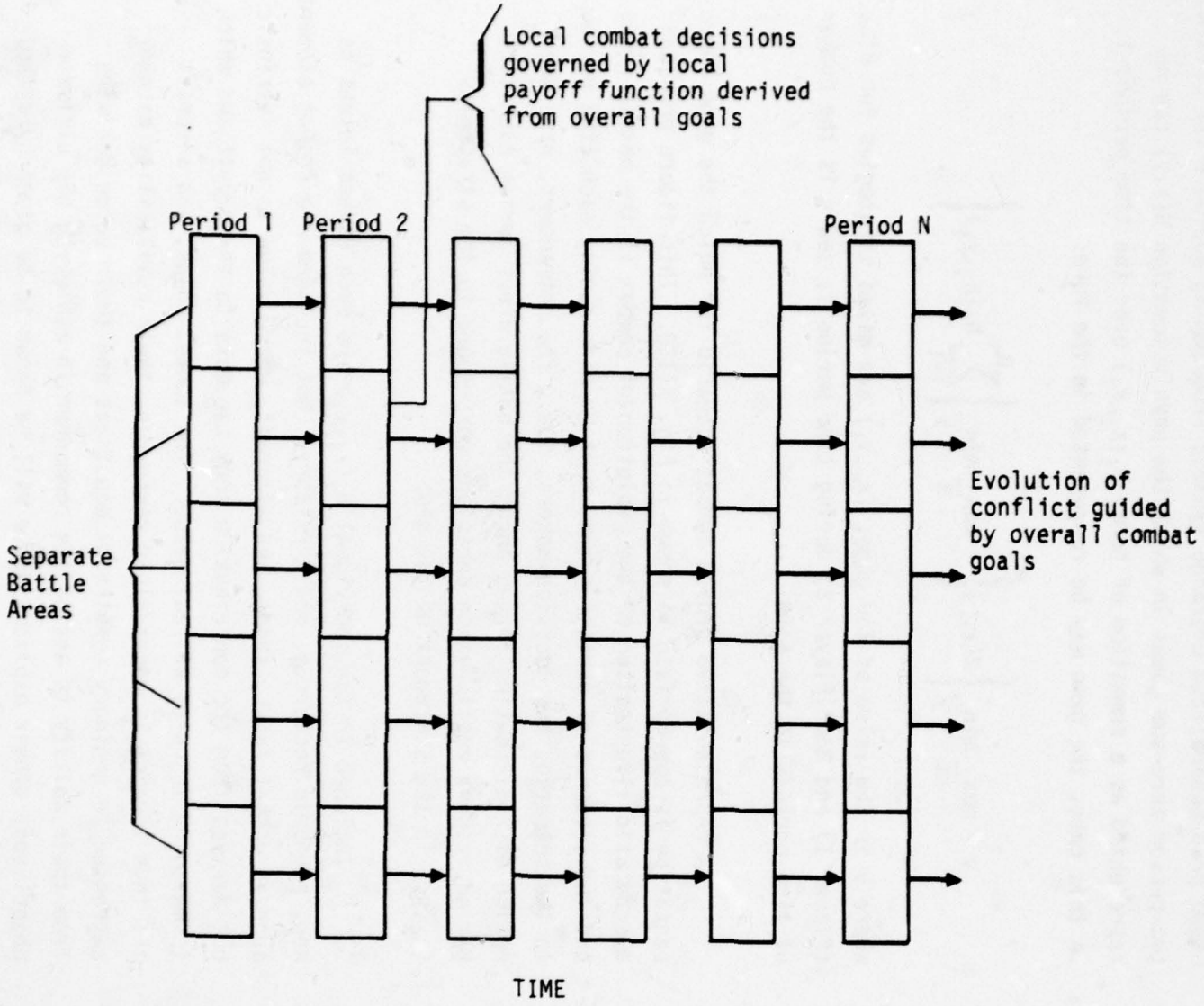


Figure III.8. Basic Concept for Ground Combat Decision Problem

ultimate objectives. These shadow values will then be shown to provide the appropriate heuristic values for use in governing the decisions of the commander during each time period. In more complex problems, where the rigorous theory cannot be applied, the interpretation of the shadow values can still be invoked to provide a heuristic procedure for generating acceptable solutions.

Before describing the Lagrange Dynamic Programming approach in detail, it is important to clarify the relationship between the local strategies (x_i, y_i) and the resource constraints. In any feasible solution of this problem, there will be very important restrictions on the strategy choices (x_i, y_i). Specifically, the allowable strategies permitted in time interval $i + 1$ must be only the ones that are feasible in view of the strategy choices actually implemented in time interval i. Typically, the strategies (x_i, y_i) that are feasible in time period i will be a function of the state of the system at the start of the period. Suppose, for example, that there are a total of M state variables that can limit the strategies x_i that are feasible for the first player in any time period. We will designate the values of these variables at the beginning of period i by R_{ij}, $j = 1, \ldots, M$. Similarly, if there are N state variables that limit the feasible strategies y_i for the second player, we designate their N values at the beginning of period i by S_{ij}, $j = 1, N$. Typically, these state variables will include at least the major resources, such as ships, aircraft, fuel, and men (as a resource), available to each side. However, depending on the level of detail of the problem formulation, they may include much more.

Each strategy x_i that might be feasible for the first player in period i will have certain requirements $r_{ij}(x_i)$ with regard to these variables. Similarly, the strategies y_i available to the second player will have requirements $s_{ij}(y_i)$. The strategies will be assumed to be feasible if, and only if, these requirements are consistent with the resources R_{ij} and S_{ij} available at the beginning of the period. The consistency requirements may be either of an equality or an inequality form. That is, the requirements for each variable j will be of the form

$$r_{ij}(x_i) \leq R_{ij} \qquad \text{or} \qquad r_{ij}(x_i) = R_{ij}$$

and

$$s_{ij}(y_i) \leq S_{ij} \qquad \text{or} \qquad s_{ij}(y_i) = S_{ij}$$

The resources R_{ij} and S_{ij} actually available in time period i, of course, are a consequence of strategy choices selected in previous time periods. We can specify these levels in a recursive form by the following equations:

$$R_{ij} = R_{i-1,j} + C_{i-1,j}(x_{i-1},y_{i-1})$$

$$S_{ij} = S_{i-1,j} + D_{i-1,j}(x_{i-1},y_{i-1})$$

Here C_{ij} represents the change in the first player's j*th* resource when the strategy pair (x_i,y_i) is implemented, and D_{ij} represents the corresponding change in the second player's j*th* resource. The equations simply reflect the fact that the resource levels at the start of the i*th* time period are equal to the levels at the start of the previous period, corrected by the change in the levels resulting from the strategy choices (x_{i-1},y_{i-1}) in the previous period. It is these equations that give rise to the Lagrange Multipliers (shadow values) for the resources.

One further simplification must be made in order to carry out the derivation. Since $(\bar{x}_i,\bar{y}_i)$ are mixed strategies the resources available at the end of a period are a function of the pure strategies that are actually played during the period. This can give rise to a totally unmanageable "tree" of potential outcomes that would have to be individually tracked if the constraints were treated rigorously. A simplifying approximation that is adequate in most contexts is to replace the actual

resource levels available at the end of a period by the expected resource levels available. Extensive discussion of the assumption is found in Part II of the Pugh-Mayberry papers.

To develop the two-sided Lagrangian formulation, we proceed by analogy with the one-sided Lagrange formulation, introducing for each R_{ij} and S_{ij} constraint Lagrange Multipliers λ_{ij} and μ_{ij}. The two-sided Lagrangian then becomes:

$$L = \sum_{i=1}^{i=n} h_i(x_i,y_i) - \sum_{i=2}^{i=n}\sum_{j=1}^{j=m} \lambda_{ij}[R_{ij} - R_{i-1,j} - C_{i-1,j}(x_{i-1},y_{i-1})]$$

$$+ \sum_{i=2}^{i=n}\sum_{j=1}^{j=N} \mu_{ij}[S_{ij} - S_{i-1,j} - D_{i-1,j}(x_{i-1},y_{i-1})]$$

Rearranging terms, we can rewrite the Lagrangian as:

$$L = L_0 + \sum_{i=1}^{i=n} L_i$$

where now

$$L_0 = \sum_{j=1}^{j=M} \lambda_{ij}R_{1j} - \sum_{j=1}^{j=M} \mu_{1j}S_{1j}$$

and where

$$L_i = h_i(x_i,y_i) - \sum_{j=1}^{j=M} (\lambda_{ij} - \lambda_{i+1,j})R_{ij} + \sum_{j=1}^{j=M} \lambda_{i+1,j}C_{ij}(x_i,y_i)$$

$$+ \sum_{j=1}^{j=N} (\mu_{ij} - \mu_{i+1,j})S_{ij} - \sum_{j=1}^{j=N} \mu_{i+1,j}D_{ij}(x_i,y_i).$$

This form exhibits the usual Lagrangian property, namely, that whenever the equality constraints are satisfied, the Lagrange payoff has the same value as the actual payoff. Therefore, for any feasible time sequence of strategy pairs, (x_i, y_i), this Lagrangian correctly represents the actual payoff.

The various terms that comprise the single-period Lagrangian payoff, L_i, have intuitive meanings that exactly parallel the meanings in the one-sided Lagrangian formulation. From the point of view of the maximizing player, the first term represents the actual contribution of the *ith* time period to the real payoff. The next two summations are value-transfer terms between time periods for the maximizing player resources. These terms carry information concerning the future value of his resources and provide a quantitative basis for allocating his resource consumption efficiently among the time periods. The first summation is equivalent to a rent he must pay for simply having valuable resources available to him during time period; the second reflects a penalty he must pay for the consumption of any resources that otherwise might be passed on for use in future time periods.

In addition to these terms (which have meanings exactly parallel to those in the one-sided LDP algorithm), there are two additional terms that have no parallel in the one-sided algorithm. The second of these terms is the significant one; it affects the maximizing player's selection of strategy. It reflects a premium that accrues to the maximizing player if he can destroy (or force the opponent to consume) resources of potential future value to the opponent.

Because of the symmetry of the expression between the minimizing and maximizing players, there is an obvious and exactly parallel (except for a change of sign) interpretation of the terms for the minimizing player.

The procedure for solving the Lagrangian for the two-sided formulation is similar to the procedure for solving the one-sided Lagrangian. An initial set of Lagrange Multipliers (λ_{ij}, μ_{ij}) is selected and then the individual Lagrangians L_i are first minimized over the y_i strategies for a given x_i strategy and then maximized over the x_i's. Defining

$$\Theta_i(R_{ij}, S_{ij}) = \max_{x_i} \min_{y_i} L_i(x_i, y_i, R_{ij}, S_{ij})$$

$$= \max_{x_i} \min_{y_i} \left\{ h_i(x_i,y_i) - \sum_{j=1}^{M} (\lambda_{ij} - \lambda_{L+1,j})R_{ij} + \sum_{j=1}^{M} \lambda_{L+1,j} C_{ij}(x_i,y_i) \right.$$

$$\left. + \sum_{j=1}^{N} (\mu_{ij} - \mu_{i+1,j})S_{ij} - \sum_{j=1}^{N} \mu_{i+1} D_{ij}(x_i,y_i) \right\}$$

The solution is obtained by calculating the $\Theta_i(R_{ij},S_{ij})$ sequentially for $i = 1$ to $i = n$, subject to the intraperiod constraints:

$$r_{ij}(x_i) \leq R_{ij} \qquad \text{or} \qquad r_{ij}(x_i) = R_{ij}$$

$$S_{ij}(y_i) \leq S_{ij} \qquad \text{or} \qquad S_{ij}(y_i) = S_{ij}$$

and the interperiod constraints:

$$R_{ij} = R_{i-1,j} + C_{i-1,j}(x_{i-1},y_{i-1})$$

$$X_{ij} = S_{i-1,j} + D_{i-1,j}(x_{i-1},y_{i-1})$$

In the expression for Θ_i, the terms in R_{ij} and S_{ij} may be omitted. In obtaining the optimal strategies, these terms are independent of the strategies (x_i,y_i) and thus do not effect the optimization. Similarly, L_0 need not be included in the optimization. We shall use the notation $\Theta_i'(R_{ij},S_{ij})$ when these are omitted in $\Theta_i(R_{ij},S_{ij})$. In this revised form, the Lagrangian is very similar to that found in the one-sided problem.

grow exponentially in value. The impact of this growth is that the multiplier tends to completely dominate the intrinsic values $h(x_i,y_i)$. The individual Lagrangian for time periods $i << n$ then reduces to

$$L_i = \sum_{j=1}^{M} \lambda_{L+1,j} C_{ij}(x_L,y_L) - \sum_{j=1}^{M} \mu_{L+1,j} D_{ij}(x_i,y_i)$$

and the decision as to what strategy to play becomes a function solely of the Lagrange Multiplier. The Lagrange Multipliers thus assume the role of heuristic values. This is the result we stated earlier.

We now consider the application of the Lagrange Dynamic Programming theory to MUSTEX and TAC COMMANDER.

MUSTEX

MUSTEX, a multiple stage nuclear exchange model, provides an excellent example of the use of Lagrange Multipliers as heuristic values and of the iteration techniques used in generating the multipliers. The MUSTEX scenario consists of an initial attack by the Soviets (say) against the U.S. missile force, followed by a U.S. counterattack against the Soviet forces. A series of Soviet attacks followed by U.S. counterattacks then follows. Following this series of counterforce exchanges, a final countervalue exchange is threatened or initiated in which each side's remaining forces are launched against the other's cities.

The measure of effectiveness--the ultimate values--in MUSTEX reflect only the ratio of the countervalue damage inflicted on the cities. The destruction of the other side's missiles in earlier exchanges does not directly enter into the computation of value destroyed. Missiles have value only insofar as they can inflict damage on the other side's cities or insofar as they can destroy the other side's missiles, which are then unable to inflict damage on the first side's cities. The effective measure of worth of missiles in the counterforce exchanges is thus measured in terms of the ultimate damage they can inflict on the other side's cities or the damage they can prevent to their own cities by

Now we have yet to relate the (λ_{ij}, μ_{ij}) for time period i to the corresponding Lagrange Multipliers for the following period. This is quite simply done, for the difference $\lambda_{ij} - \lambda_{i+1,j}$ is equal to the marginal value of resource j. Assuming, for simplicity, that the necessary derivatives all exist, this may be formally proved by differentiating the Lagrangian L_i with respect to R_i and setting it to zero. This gives

$$\lambda_{ij} - \lambda_{i+1,j} = \frac{\partial}{\partial R_{ij}} \left\{ \Theta_i'(R_{ij}, S_{ij}) \right\}$$

and similarly for μ_{ij},

$$\mu_{ij} - \mu_{i+1,j} = \frac{\partial}{\partial S_{ij}} \left\{ \Theta_i'(R_{ij}, S_{ij}) \right\}$$

If we set $\Delta\lambda_{ij}$ equal to the right-hand side of the first equation and $\Delta\mu_{ij}$ equal to the right-hand side of the second equation, we have

$$\lambda_{i+1,j} = \lambda_{i,j} - \Delta\lambda_{i,j}$$

$$\mu_{i+1,j} = \mu_{i,j} - \Delta\mu_{i,j}$$

so that the marginal value of a resource within a given time period provides the means for relating the Lagrange Multiplier between time periods.

The resulting iteration procedure then consists of the following steps:

1. Assume initial values for all Lagrange Multipliers, λ_{ij} and μ_{ij}.
2. Using these values, solve the min-max games defined by $\Theta_i'(R_{ij}, S_{ij})$ starting with i = 1 and moving forward in time to i = n. Use the initial resource levels R_{ij} and S_{ij} in

the first period and the R_{ij} and S_{ij} interperiod constraints to determine the initial resource levels in subsequent periods.

3. During Step 2, calculate and record the marginal values of resources R_{ij} and S_{ij} in all time periods i using the $\lambda_{ij} - \lambda_{i+1,j}$ and $\mu_{ij} - \mu_{i+1,j}$ equations.
4. If convergence is adequate, stop; otherwise, go to Step 5 and recycle.
5. Starting with time period n and working backward in time, calculate "improved" estimates of the multipliers using the $\lambda_{ij} - \lambda_{i+1,j}$ and the $\mu_{ij} - \mu_{i+1,j}$ equations. (The final λ_{ij}'s and μ_{ij}'s are either zero or are readily calculable.)

In general, the convergence criteria requires that the Lagrange Multipliers agree in value from iteration to iteration. Practically, the formulation of the criteria poses no problem but in general must be tailored to each application. Further discussion of the criteria formulation is given in Part II of the Pugh-Mayberry papers.

Before discussing the application of the Lagrange dynamics programming theory to MUSTEX and TAC COMMANDER it is important to clarify the interpretation of the Lagrange Multipliers in dynamic simulations and to indicate one of their properties that is of considerable significance in the design of value-driven decision systems--namely, the role of Lagrange Multipliers as a heuristic value.

The Lagrange Multiplier $\lambda_{i=1,j}$ represents the value of resource j in time periods i + 1 onward. As an example, for an aircraft, λ_{ij} represents the value of using the aircraft during periods i + 1 forward, so that it represents the cost that period i would be required to pay future periods should the aircraft be destroyed in an *ith* period mission. Alternatively, λ_{ij} represents the price future periods would be willing to pay period i for the aircraft. One implication of this result is that the value of the Lagrange Multipliers tends to increase as one moves backward in time. In fact, in many examples, the multipliers are found to

destroying the other side's forces. A convenient measure of this "derived" value is the increase in the ultimate value that could be destroyed if an additional weapon--i.e., a missile--were added to the inventory. However, this is just the shadow value (Lagrange Multiplier); it provides a measure of the worth of the weapon to future time periods. Shadow values thus assume the role of heuristic values governing the assignment of missiles during the counterforce exchanges.

The formal derivation of the role of the shadow values follows immediately from the form of the individual Lagrangian, which reduces in MUSTEX to

$$L_i = h_i(x_i,y_i)S_{1,last} + \sum_{j=1}^{M} \lambda_{i+1,j} C_{ij}(x_L,y_i)$$

$$- \sum_{j=1}^{N} \mu_{i+1,j} D_{ij}(x_L,y_i)$$

The delta function δ_{ik} is equal to 1 for i = last and zero for $i \neq$ last. The C_{ij}'s and D_{ij}'s are equal to the number of missiles of each side expended or destroyed in the engagement. Since only one side attacks at a time, the C_{ij}'s will be the number of missiles expended and the D_{ij}'s the number of missiles destroyed if Blue is attacking and the C_{ij}'s the number of destroyed and the D_{ij} the value expended if Red is attacking. Following each strike it is assumed in MUSTEX that each side has perfect information on the status of his own and of the opposing forces, so that the games reduce to one of complete information. Only pure strategies need then be considered.

To solve for the force allocations, we must first define the value function $h_i(x_i,y_i)$ in somewhat greater detail. We may define it either as the difference between the damage inflicted by Blue and Red or as the ratio of the two damages. For simplicity, in the ensuing discussion, we will use the difference in damage incurred for $h(x_i,y_i)$. The damages themselves will be assumed to be normalized to one, so that a damage level of 1.0, for example, corresponds to destruction of the entire countervalue target base.

MUSTEX allows for multiple missile types and for MIRVs. Standard Operations Research procedures are used for developing equivalences among missiles on the same side. We will therefore develop the force allocation argument assuming only one type of missile on each side, referring to multiple missile types only to indicate how they affect the detailed assignments. The basic scheme is to develop an initial estimate of the Lambdas (Lagrange Multipliers) for the first stage; then to estimate the Lambdas for successive stages by allocating the forces during each stage; and then to compare the last counterforce Lambdas with the marginal countervalue payoffs which we can calculate explicitly. If these fail to agree, we can projet the marginal countervalue payoffs--which represent the new estimate of the final Lambdas--back to the first stage using the transition relationships obtained in progressing forward. (In the course of moving forward, we calculated and retained the relationship between λ_{ij} and $\lambda_{i+1,j}$. We simply invert the relationship to move backwards.) This provides us with new initial estimates of the λ_{ij}'s. We then proceed forward once again repeating the whole process. After a number of iterations--generally from six to fifteen--the final Lambdas calculated moving forward agree closely with the marginal countervalue payoff calculated directly. The solution has then converged.

To implement this process, we must describe how the marginal countervalue payoff is calculated and how the Lambdas are modified from stage to stage. Let us first consider the computation of the marginal countervalue payoff. Assume for a moment that the nuclear exchange consists solely of the final countervalue change and furthermore, as stated above, that the payoff function is the difference of the damage inflicted by Blue minus the damage inflicted by Red. We will make one modification to the above discussion; we will direct our attention to the exchange ratios between Blue and Red missiles, i.e., the number of Blue missiles that Blue is willing to expend to destroy one Red missile. This corresponds to looking at the ratio of the Lagrange Multipliers and is all that is required to make the missile assignments.

Now one typically has available damage curves of the form shown in Fig. III.9 which plots the damage that Red and Blue weapons (missiles) can inflict against the adversary countervalue target base as a function of the number of weapons in the inventory. In Fig. III.9, this damage, normalized to 1.0, is plotted as a function of weapon inventory for both Red against Blue and Blue against Red. The dotted lines in the figure correspond to initial Blue and Red inventories. Associated with these lines is an indication of the additional damage that would be effected if additional Red and Blue weapons were added to the inventory. It is seen that it takes three additional Blue weapons to inflict as much damage (expressed as a percentage of target base) as one Red weapon. Since the payoff is the difference between the damage inflicted by Blue and the damage inflicted by Red, the exchange ratio is three Blue weapons to one Red weapon. (If we were interested in the values of the Lagrange Multipliers rather than in the ratio, we would look at the actual damage inflicted by the additional weapons rather than just the ratio of the damages.)

Note one very important feature of Fig. III.9. The exchange ratios--as well as the absolute damage--depend on the initial inventories of weapons. If the inventories change, the exchange ratios also change. Since the inventories remaining at the end of the counterforce exchange are not known in advance, the exchange ratios are also not known. They must be obtained by an iterative procedure. For the first iteration, the initial set of shadow values for each stage are constructed as though the stage represented the last counterforce attack. The weapon inventories that are assured are those remaining after the completion of the immediately preceding attack. All initial exchange ratios are thus obtained directly from the countervalue curves, using the initial inventory for the weapons available in the first stage, those remaining after the completion of the first stage attack for the second stage, and so forth. Using these exchange values the first iteration can thus be carried out. Exchange values for subsequent iterations can then be determined by

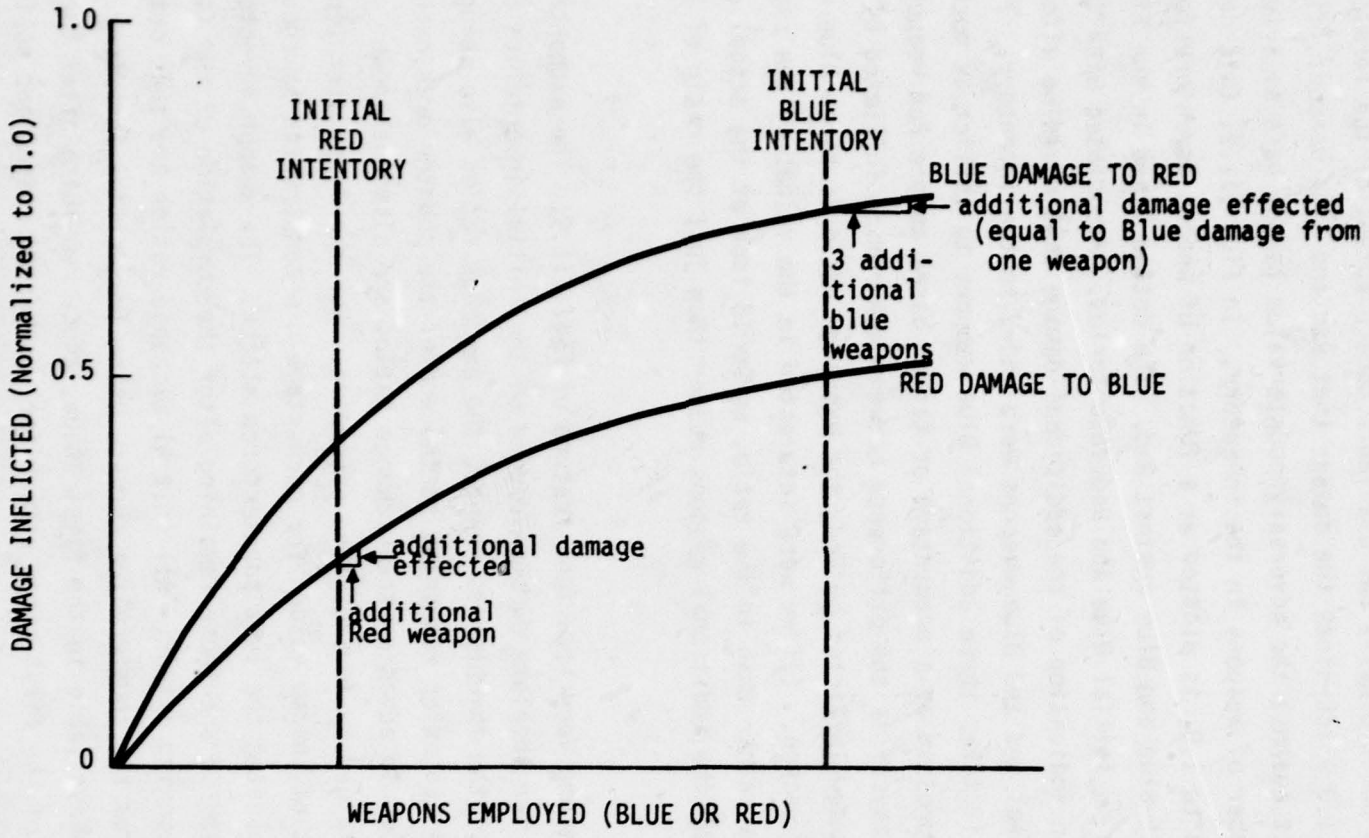


Figure III.9. Countervalue Damage Inflicted by Blue and Red Weapon Inventories

backtracking, starting from the exchange ratios determined from the countervalue damage function using the final counterforce missile inventory obtained in the last iteration.

The final step necessary to complete the specification of the allocation algorithm is the computation of the change in the exchange ratios from period to period. For this purpose, assume we are developing the laydown for the final counterforce attack-say Blue on Red--before the countervalue exchange. Using the shadow values, we allocate the Blue missiles to the Red targets (missiles) using the difference between the shadow value of the Blue missile and the designated Red target as the value associated with making a particular assignment. (In the model, P_k's and MIRVing complicate but do not affect the basic concepts involved.)

Now consider a Blue missile type for which all of the missiles were not used during the period. Adding an additional missile of this type to the inventory obviously would not increase the value of the attack. The missile's worth is measured totally in terms of its future value in the countervalue attack. Hence, its Lagrange Multiplier, or more precisely its exchange ratio, will not change as a consequence of the attack. On the other hand, suppose we exhaust a particular missile type in the attack. Adding another missile of that type might then allow a more valuable target to be attacked--a target that could do, say, three times the damage of the attacking missile to the countervalue target system. In this case, the exchange ratio of the attacking missile should be increased, for its contribution to the final payoff is three times greater than had previously been anticipated. Thus, in backtracking to find the value of the exchange ratio in the preceding period, i.e., the i - 1 period, the countervalue exchange ratio should be enhanced to reflect the value of the missile during the current time period. On the other hand, suppose a Red missile type (target) is exhausted during the attack. If an additional target were present, it could then possibly be destroyed by a lower "value" attacking missile. Its worth is then

less than had been previously estimated. Its value therefore must be decreased in obtaining its value for preceding time periods since it cannot achieve the countervalue capability it would have were it to survive.

By similar procedures the exchange ratios are calculated for all stages. The computation of the exchange ratios and hence, the allocation of forces and the value of the game can then be readily determined.

TAC COMMANDER

The TAC COMMANDER model provides a particularly simple example of the use of Lagrange Multipliers as heuristic values. It illustrates that the rather complex Lagrange concept can be applied in a very simple way and still provide useful results.

The problem considered in TAC COMMANDER is the assignment of strike aircraft to time-perishable targets. The targets fluctuate substantially in value and their movement makes their redetection by strike aircraft very difficult if any significant period of time elapses between detection--generally by reconnaissance aircraft--and arrival of the attack aircraft in the target area. The effective target damage associated with assigning aircraft to the target may be represented in the form

$$v = v_{int} * P_s \, e^{-[(t_a - t_0)/\tau]} \cdot P_k$$

where v_t is the intrinsic, i.e., time-dependent, value of the target; P_s is the probability that the attacker reaches the target; t_a is the attacker's arrival time at the target; t_0 is the target observation time (by the reconnaissance aircraft); τ is the mean redetection life of the target; and P_k is the probability the attacking aircraft destroys the target.

The exponential decay value makes it imperative that attack aircraft are available for near immediate assignment. Assuring the availability of the strike aircraft is particularly difficult for the targets are detected essentially at random, and the variability in intrinsic value of targets, coupled with a limited availability of strike aircraft, makes the decision whether to assign an aircraft to a target or to withhold it for a more valuable target a very critical one. In the original TAC COMMANDER model, the decision was made by means of a set of priority rules (decision rules). The priority rules specified the actions to be taken under a variety of conditions. "Close air support targets," for example, were always assigned the highest priority and generally--when aircraft were available--attacked immediately. The priority scheme worked reasonably well but lacked the flexibility or the sensitivity to perform the finely timed allocation that was required to confidently evaluate the forces. A simple value scheme was then tried. The following form was selected--heuristically--for the individual Lagrangian L_i:

$$L_i = v_i - P_{des} * v_a - \frac{N_0 R_0}{(N_a + N_0)/2}$$

where the Lagrangian is evaluated for a specific aircraft located on a designated airbase. The first term in the Lagrangian is the time-dependent value of the target as indicated above. The second term reflects the value of the aircraft to future time periods. It represents the target value that would not be destroyed in the future if the aircraft were destroyed on a mission during the current time period. Equivalently, it reflects the price the future is willing to pay to have the aircraft at its disposal. It thus corresponds to the "dynamic" shadow value of the aircraft. Since the aircraft is not necessarily destroyed on the mission, the shadow value must be multiplied by the probability the aircraft is destroyed on the mission. The second term thus consists of two factors: the shadow value v_a and the probability that the aircraft is destroyed during the mission P_{des}.

In TAC COMMANDER, the dynamic shadow values for the various attack aircraft are selected by the user; no iterative procedure is used for selecting them. They reflect his personal estimate of the worth of the aircraft.

There are three major effects that the dynamic shadow values have on the allocation process. First, if the probability of the aircraft being killed in attacking a target is high, no aircraft will be assigned to the target unless the target's value is unusually high. The mission is not worth the risk of the aircraft. Second, if two aircraft of different types are available on a given airbase for assignment to a mission the least valuable aircraft will be assigned (if the risk to each aircraft is the same). And finally, the aircraft will be selected for assignment to a target from an airbase in which the risk of the aircraft being destroyed is minimum.

The form of the dynamic shadow value term also results in an implicit trade-off being made among the factors that contribute to the assignment of an aircraft to a mission. For example, in the selection of the airbase the time of arrival is balanced against the probability the aircraft survives the mission. If the arrival time is extended, the value v will diminish, which will tend to offset a high probability of survival (a small value of P_{des}).

The third term in the Lagrangian also represents a shadow value, but in this case a "local" shadow value which is not related to the future worth of the aircraft but rather to the fact that limited resources--aircraft--are available during the time period. It is analogous to the situation that exists in MUSTEX during a single stage. In that case, it was required to assign a limited number of resources to targets in an optimal manner (for a given set of dynamic shadow values). The assignment problem was solved by using the Generalized Lagrange Multiplier approach. Lagrange Multipliers were introduced to replace the constraints

on the missiles; these multipliers could be interpreted as the increase in target value that would be destroyed during the given time period by the addition of one more missile of the specified type. In the same way, the shadow values in TAC COMMANDER represent the value that could be realized if one additional aircraft of the specified type were available during the current time period.

The specific form of the local shadow value term is constructed by analogy with the procedure used in the Merck Scheduler, in which an interplay is introduced between the instantaneous--i.e., local--shadow values and the time-averaged shadow values. As in the Scheduler, the time arranged shadow values in TAC COMMANDER, R_0, are formed by taking a weighted average over the instantaneous shadow values for previous periods. The shadow values thus vary slowly with time tending to increase as the number of target detections increase. This has the effect of making the rent on an aircraft higher, for on the average it is more likely that an aircraft will be unavailable when a valuable target is detected.

The instantaneous shadow values are constructed by first defining a desired resource level N_0, which represents the number of aircraft that a commander would ideally like to maintain on alert status for assignment to targets. A ratio between N_0 and the average number of aircraft available before and after an aircraft is assigned--N_0 and N_a--is then constructed and used as a multiplicative factor on R_0. This defines the instantaneous shadow value. The instantaneous shadow values thus correspond to the entire third term of the Lagrangian. Its effect is to increase the rent on an aircraft type that is "momentarily" in high demand. It thus exhibits the behavior desired.

TAC COMMANDER illustrates that the concepts of the Generalized Lagrange Multiplier theory and, in particular, the shadow values can be successfully applied in a wholly heuristic manner without recourse

to the intricacies of the formal theory. In this form the theory has been successfully and effectively applied in the development of many simple models.

D. VALUE GENERATION FOR LARGE SYSTEMS

The generation of values for large systems is in many cases considerably simplified by the character of the solution space for the system. Typically, many solutions are found that are arbitrarily close to the "optimal" solutions. These solutions frequently display properties that are considered desirable in a real-world solution but which have not generally been taken into account in generating a computer solution, for the usual procedure of representing them by rigid constraints greatly complicates the solution procedure. These solutions can frequently be found, however, by adding to the value function a small "symmetry breaking" term whose effect is to remove the "degeneracy" among the solutions and to identify the solutions with the desired property. The specific form of the symmetry breaking is largely arbitrary, illustrating the quite general principle that as long as the value functions exhibit certain general features its detailed structure is ordinarily not critical to the determination of a satisfactory solution.

These properties are most simply illustrated in the 44 City Study[1] and more specifically in the development of a student busing plan for Prince George's County, Maryland. Prince George's County, under court order to integrate their schools, sought a busing plan that would minimize the number of busing miles required, subject to constraints on the racial composition and the capacity of the schools, and to certain subsidiary constraints on, for example, the maximum distance any student would be bused.

---

[1] H. Everett, G. Pugh, School Desegregation with Minimum Busing, Lambda Paper 68, December 1976.

The construction of the busing plan required as a first step the determination of the distances of the students from the schools in the system. This was treated by dividing the county into "block groups," each encompassing an area containing about 150 students. (The total student population was about 150,000 distributed over approximately 250 schools.) Using road maps to establish distances and census data to localize the student population, the distances of each of the block groups from each of the schools were then calculated using a modified minimum path algorithm. These distances then served to define the alternative courses of action for assigning students to school. By using the algorithm courses of action that were obviously non-optimal were eliminated from further consideration.

For the purpose of making assignments to schools, students were classified as minority and non-minority students. The objective was then to achieve as equitable a balance as possible beyond the minority and the non-minority students in each of the schools in the county. This was achieved by using an additive value function over schools, where the value function for each school had the form

$$V = Sp(1 - p)$$

where S is the total student population of the school and p is the fraction of the student population that is minority. The expression has the property that it is zero if the school is either all minority or non-minority and is maximum if the school contains equal numbers of minority and non-minority students. The expression also has the very important property that it is convex; this insures that the optimal solution has the property that all schools will have a balance between minority and non-minority students rather than some schools being very well balanced and others not balanced at all. This property was the reason for using the nonlinear objective function. For a linear function, this balance would not have been obtained.

The integration of the students in the schools was to be achieved in such a manner that it minimized the number of miles students were bused. This requirement is ambiguous in that it requires two incompatible objectives to be simultaneously optimized. The requirement therefore had to be interpreted. The way in which it was done illustrates the character of solutions in large systems.

As an initial attempt to develop an integration plan, the value function V was maximized subject to a rigid constraint that permitted no student to have more than a 30-minute bus ride. (Average speeds were specified for the roads in the county, so that busing times could be converted into distances.) This produces a high level of desegregation but resulted in extensive busing of students.

To improve the solution, i.e., to reduce the number of miles bused, it was necessary to include a distance penalty in the solution procedure. The solution procedure had to recognize that it was less desirable to bus a student 10 minutes than to bus him only 5 minutes. The penalty was introduced not as a rigid constraint--by reducing the allowed travel time to 10 minutes, for example--but instead, by subtracting from the value function V a small penalty term for each student, proportional to the distance that he is bused.

The addition of the small penalty term to the value function greatly--by a factor of several times--reduced the number of student miles bused and did so with only a slight decrease in the level of desegregation achieved. The small penalty term had, in effect, broken the symmetry among the set of near-optimal solutions and selected one that satisfied the subsidiary constraint.

One additional modification was made to the value function. By introducing a linear penalty on the distance each student was bused, the actual effect was only to penalize the total mileage of all students bused and not to penalize each student individually. Although this

procedure reduces the total number of miles all students, as a group, were bused, individual students in the busing plan were frequently bused up to the 30-minute limit. The plan was thus not yet satisfactory.

To reduce the distances over which individual students were bused, the linear distance penalty was replaced by a quadratic one. This procedure separately penalized the value function for the distance each student was bused. Employing the quadratic penalty, a new solution was found, which once again achieved only a slightly lower level of integration than the original plan and which tended to equalize the distance over which individual students were bused.

The form of the value and penalty functions illustrates the general point made at the beginning of the section--namely, that the detailed form of the functions is not important as long as certain general features are maintained. The important features of the value function V are its convexity and its limiting values which discourages either wholly minority or majority schools and which encourages a balance between them. For the penalty function the important feature is that it individually penalizes the value function for the distances individual students travel. These features are all satisfied by simple quadratic functions. This forms the basis for using standardized functions in value-driven theory.

IV. ADVANCED CONCEPTS IN VALUE-DRIVEN DESIGN

In the last chapter a general introduction was given to the design of values in value-driven decision systems. The discussion was intended to provide the prospective designer with a feeling for developing values in a value-driven system and with general guidelines for use in the development. With the possible exception of the use of Lagrange Multipliers, however, it was quite clear that a considerable amount of ingenuity would be required of the designer in developing an acceptable value structure. Fortunately, for a rather wide class of combat applications much more precise guidelines can be given. These will be developed in this chapter using the TAC BRAWLER many-on-many air combat model as an illustration of the application of the general design principles.

A. GENERAL CONSIDERATIONS IN VALUE DESIGN

As was emphasized in the discussion of the Merck Scheduler, the purpose of the values in value-driven decision theory is solely to provide a means for selecting a preferred course of action from among a set of alternatives. Thus, the only criterion that must be met in value design is that the resulting value structure have this property. This provides the designer with considerable flexibility in how he structures his value functions. Typically, he will adopt a measure of merit or design principle such as the profit maximization in the Scheduler, to use as a guide in designing the value function, but he then will deviate from it, such as in establishing the relationship between the minimum acceptable and normal minimum inventories in the Scheduler, either for reasons of simplicity or because the display of the desired behavior by the simulation necessitates it.

The use of scalar value functions has provided an adequate characterization of value in all applications that have so far been considered. No difficulty in trading off the relative utility of seemingly quite different types of considerations has been found. This is partially

a consequence of using multiple level mental models, each of which generally requires only the consideration of attributes that do not differ substantially from one another in kind. For example, in TAC BRAWLER at the pilot tier level a pilot contemplating an attack on a hostile aircraft must put relative weights on such attributes as his specific energy relative to the hostile aircraft and his rate of closure on the aircraft. These types of trade-offs are relatively easily made.

A very general and important observation that has been made in the development of values for value-driven decision systems is that the detailed form of the value function is not critical to the selection of alternatives. As long as the value function has the correct general form--proper behavior in the limits, proper slopes, etc.--a satisfactory alternative will be selected. This is particularly true for many combat applications (the Scheduler is rather special in this regard). A consequence of the observation is that standard forms can be defined for constructing value functions. Parameters within the forms are adjusted for a particular application. The standard set of forms and illustrations of their applications are presented in Chapter II.

Another consequence of the observation is that the additive value function

$$v = \sum_i \alpha_i v_i$$

can frequently be used either alone or as part of a more comprehensive expression. In this expression the individual v_i's are value components--such as the relative specific energy and closing velocity of the attacking aircraft cited above--and the α_i's are appropriate weighting factors. To use the additive expression, the v_i must, of course, represent essentially independent contributions to the value. If an attribute depended on a second attribute to realize its value then the form would be inappropriate. Such a case would occur, for example, in specifying the conditions that must exist for an attack aircraft to launch a missile against a hostile

aircraft. The attack aircraft must be at least a given distance behind the hostile and within a specified altitude window. This type of situation frequently arises in the calculation of the individual v_i's. In that case a product form for representing the v_i's is nearly always adequate.

Two constraints are generally imposed upon additive value functions. The sum of the weights α_i are normalized to one and the value components are constrained to lie in the interval -1 to +1.

$$\sum_i \alpha_i = 1 \qquad \text{and} \qquad -1 \leq v_i \leq +1$$

The constraints on the α_i's are primarily for convenience, intended merely to simplify the tasks of assigning weights to the various attributes. The normalization of the v_i's is more fundamental. It is introduced to ensure the component values remain bounded. One component will thus never completely dominate all others. For most cases of practical interest this is the desired situation. For example, for a friendly contemplating an attack on a hostile, a tail approach is generally considered to be the most desirable position, whereas the converse situation with the two aircraft interchanged the least desirable. Other considerations are also important, however, and the position considerations alone should not be allowed to completely dominate all others. This "balance" among components is maintained by using the bounded value functions.

A particularly powerful technique that has been extensively exploited involves the use of "surrogate probabilities" as a means for replacing real probabilities--for example, probabilities of kill--by valuative considerations that reflect the likelihood that the action to which the probability refers will be realized. In the selection of alternatives, it is frequently used for dealing with probabilities that are difficult to determine precisely. It is best illustrated by an example.

In TAC COMMANDER a pilot is frequently required to make a determination of which of several aircraft to attack. In making the determination he weighs the value of the target--which involves a number of factors that may vary dynamically with the situation--the probability of killing the target, his own value and the probability that he will be killed. His value function then assumes the form:

$$v = v_{hostile} * P_{k_{hostile}} - v_{self} * P_{k_{self}}$$

The determination of the P_k's, if performed rigorously, would require an extensive set of very detailed computations. For the purpose of selecting a desirable alternative such calculations are not only overly complex but also are unnecessary. Since only the relative desirability of alternative courses of action are being compared, it is sufficient to replace the P_k's by surrogate quantities that reflect the relative desirability of the attacker's position vis-à-vis the potential target. It is also desirable, of course, that the probabilities are bounded, but this is taken into account by the conditions imposed on the additive value function. (All quantities are positive, so that the surrogate probability is positive.)

The quantities that are important in evaluating the relative advantage of the attacker over the target, assuming equivalent aircraft, are the relative position of the aircraft, the rate of closure of the two aircraft, the relative specific energy and the visibility. With these considerations the surrogate probability may be written as:

$$p^{surrogate}_{k_{hostile}} = \sum \alpha_i v_i$$

where

$$\sum_i \alpha_i = 1, \quad 0 < v_i \leq 1$$

and

v_1 evaluates the worth of position
v_2 evaluates the worth of closure rate
v_3 evaluates the worth of specific energy
v_4 evaluates the worth of visibility

The function v_i are constructed from kinematic and dynamic consideration of the capabilities of the two aircraft. The weighting factors α_i are either selected by the user or a set of values may be used which were selected by the designer to insure that the aircraft exhibits reasonable behavior.

By the use of the surrogate probabilities, which provide a weighting of the factors contributing to successful probability of kill, the alternative courses of action may be compared and an alternative selected for implementation without ever explicitly calculating the probabilities of kill. The surrogate probability approach is thus seen to be very powerful and very general. It is used extensively in the development of value-driven systems.

The explicit determination of the value $v_{hostile}$ and v_{self} will be discussed in Chapter VI. The determination of their value depends on the pilot's conception of his and the target's situations and thus relates to the situation perception activity.

In the next section the development and application of standardized forms for value functions is discussed. The use of value-mediated command languages is then described as a means for communicating preferences between decision entities and between different levels of the same entity.

B. STANDARDIZED FORMS FOR VALUE FUNCTIONS

As indicated in the last section, from the perspective of alternative selection, the specific form that is selected for a value function is not

critical as long as certain general features relating to behavior in the limit, monotonicity, and so forth, are maintained. Fine tuning of the value functions is accomplished through the adjustment of parameters which define the functions.

In value-driven decision theory two elementary functional forms and two supplementary forms that are simple modifications of the two basic types are defined. The forms are known as the Border, the Cauchy, and the Reward functions, type I and II.

1. The Border Function: Border (x_1, x_s). The form of the Border function is illustrated in Fig. IV.1. The algebraic form of the Border function is given by

$$Border\ (x, x_s) = \begin{cases} \dfrac{1}{\left(1 + \dfrac{x}{x_s} - 1\right)^2} & x < 0 \\ 1 - \text{Border}(-x, x_s) & x > 0 \end{cases}$$

where x_s is a scaling factor greater than zero.

The Border function is used in situations where it is desirable to have x greater than some value x_0. It is thus commonly used in the form $B(x - x_0, x_s)$. The scaling factor x_s provides a measure of the characteristic distance over which changes in x are significant. In the limit of $x_s \to 0$ the Border function becomes a step function.

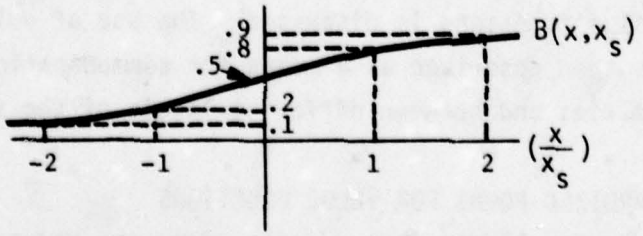


Figure IV.1. Standard Form of the Border Function $B(x, x_s)$

An example of a situation in which Border functions are particularly useful is in defining missile launch envelopes, which are used to specify regions of space from which a missile launched at a target is likely to be successful in killing it. If the decision is to be considered by a pilot--or more precisely, by a decision element representing the pilot--then it is appropriate to define a value function which is large in the launch envelope and small outside of it. If we restrict ourselves for simplicity to two dimensions, allow x to be the (positive) distance from the tail of the target to the attacker and θ the (positive) line-of-sight angle of the target with respect to the longitudinal line of the attacker, then we can define a launch envelope in which the attacker must be in a window extending from 3000-10,000 ft behind the target and with a line-of-sight angle no greater than 30° as follows:

$$v = B(x - 3000, 1000)B(10000 - x, 1000)B(30° - \theta, 5°)$$

In the expression we have provided a leeway of 1000 ft in the definition of the launch envelope and a leeway of 5° in the line of sight.

From the perspective of value design, the use of the Border function in defining the launch envelope has a number of advantages over the corresponding step function that is classically used in defining launch envelopes. These are:

- It provides a much more physical representation of the launch envelope. Launch envelopes are not sharply defined regions. Typically the probability of kill varies slowly in space and the value function should be designed to reflect this variation. Moreover, the use of Border functions avoids the common problem encountered in simulation design in which the output varies discontinuously depending on whether a trajectory passed in or out of the launch envelope.

- The attacker is rewarded most highly for being in the center of the envelope. A step function makes it as desirable to be on the edge of the envelope as in the center.
- Finally, and most importantly, for value function design the presence of the finite slope of the value function encourages the pilot to move his aircraft toward the center of the launch envelope. This is particularly important in a dynamic environment in which the pilot has the opportunity to maneuver his aircraft into a more desirable position for launch.

The Border function is thus seen to have features that make it a particularly attractive function to use in value design. It is used extensively in the applications of value-driven decision theory.

2. The Cauchy Function. The form of the Cauchy function, which is used extensively in physics and in statistics, is shown in Fig. IV.2. Its algebraic form is given by:

$$C(x, x_s) = \frac{1}{1 + \left(\frac{x}{x_s}\right)^2}$$

where x_s is a scaling factor as before.

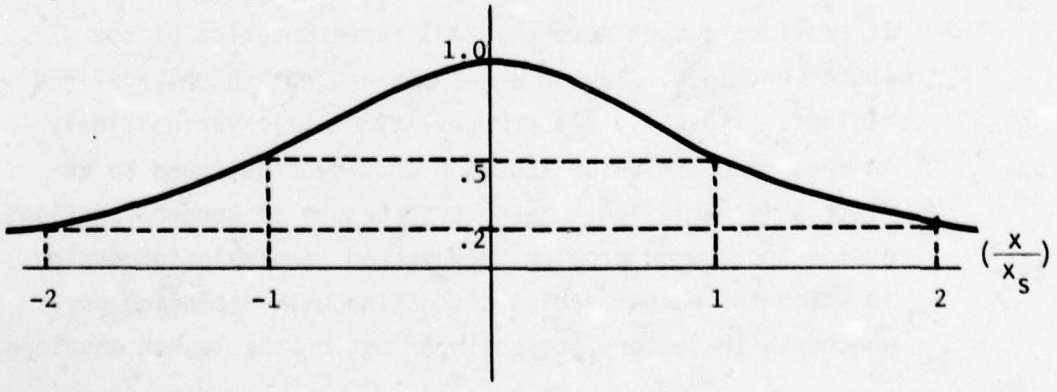


Figure IV.2. Standard Form of the Cauchy Function $C(x, x_s)$

The Cauchy function is most frequently used in a situation in which it is desirable to have two quantities assume nearly the same value. As in the case of the Border function, it is therefore typically used in the form $C(x - x_0, x_s)$. A typical application of function in TAC BRAWLER is in "formation-keeping," i.e., one aircraft, typically the wingman, must approximately maintain a given position with respect to the flight leader. In a one-dimensional approximation, x_0 is set to the desired position of the wingman. A quadratic "penalty" is then associated with variations from the desired position, with x_s reflecting the severity of the penalty.

The Cauchy function is also frequently used in situations in which the penalty associated with variations in x are not symmetric in the $\pm x$ directions. A typical example occurs in TAC BRAWLER in the treatment of visibility. Ideally, an attacker wishes to have a target positioned (with some qualifications) directly in front of his aircraft. From the viewpoint of visibility, variations in the target's position vertically upward are not particularly significant. The target could be lost in the sun but otherwise the effects on visibility are small. On the other hand, variations vertically downward are very significant; the target could be lost below the nose of the aircraft. To treat this situation the Cauchy function is still used but different values of x_s are used for $\pm x$. Since the Cauchy function has a value of one at the origin independent of x_s no discontinuities in the functions are encountered. This simple generalized treatment of the Cauchy function is found to work successfully in many applications.

3. The Reward Functions, Types I and II. The form of the two Reward functions are shown in Fig. IV.3. The analytical forms of the Reward function are given by:

$$R_1(x_1, x_s) = 2[B(x_1, x_s) - 1/2]$$

$$R_2(x_1, x_s) = 2[C(x_1, x_s) - 1/2]$$

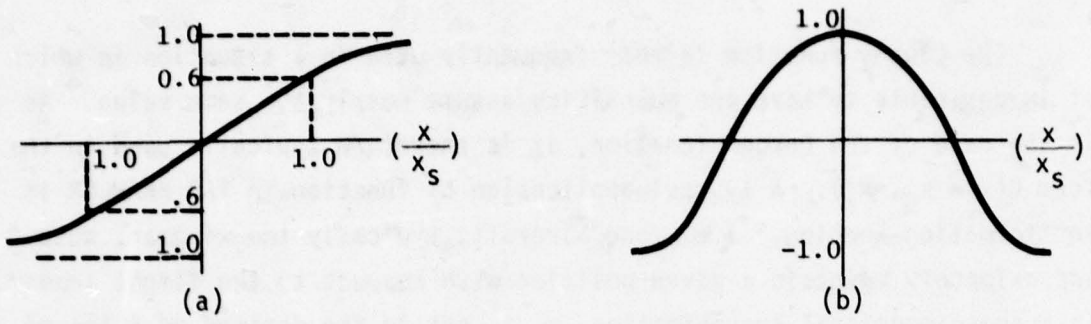


Figure IV.3. Standard Form of the Reward Function, Type I (a) and Type II (b).

The Type I Reward function is thus seen to be directly related to the Border function and the Type II function to the Cauchy function. The Reward functions are used primarily in the construction of additive value functions in which a particular attribute (value component) is judged to be sufficiently critical that in evaluating an alternative in which the attribute scores poorly it is necessary not only to assign the attribute a near zero value but to actually make it negative. It then not only does not contribute to the value function but also diminishes the contribution of the other attributes. A typical situation in TAC BRAWLER in which the Reward functions are used in the case in which a defender has long-range air-to-air missiles and the prospective attacker does not. In this situation other conditions must be extremely favorable before a decision is made to initiate an attack. The Type II Reward functions also play a special role in treating a problem that arises in the treatment of bounded value functions. We now turn to a discussion of that problem.

The problem arises in the situation in which a variable (attribute) is far from its desired value. For example, for an attacker closing on a target preparing to launch an air-to-air missile against it, there is generally a preferred closing speed S*. The speed is chosen, so that the performance of the missile is optimum. Now in any given situation there is an actual closing speed S_{act} and hence a differential closing speed

$$\Delta S_{act} = S_{act} - S^*$$

Following our previous discussion the value associated with the closing speed is represented by a Cauchy function $C(\Delta S_{act}, S_{scale})$. This function is large when ΔS_{act} is small. Now suppose that the actual closing speed is far from the preferred speed S*. The situation is then as illustrated in Fig. IV.4. The figure shows that in the neighborhood of large value of ΔS_{act} the Cauchy function is flat, so that there is little difference in the values associated with good and bad courses of action (indicated by Δs_1 and Δs_2 in the figure). Thus the Cauchy function does not tend to encourage improvements in variables that are far from this optimal value. This is a general feature of bounded value functions. Unbounded functions, on the other hand, approach infinity for large values and hence tend to strongly encourage "motion" toward the preferred values of the variables.

The best solution that has been found to the problem of encouraging motion toward the preferred value of the variables in bounded value function is the addition of a small Type I Reward function to the Cauchy function. Constructed properly this will have the effect of distorting the value function in the neighborhood of the current value of the closing speed S* so as to encourage motion toward the preferred value.

To make the argument explicit, let

$$\Delta S_{act} = s_{act} - S^*$$

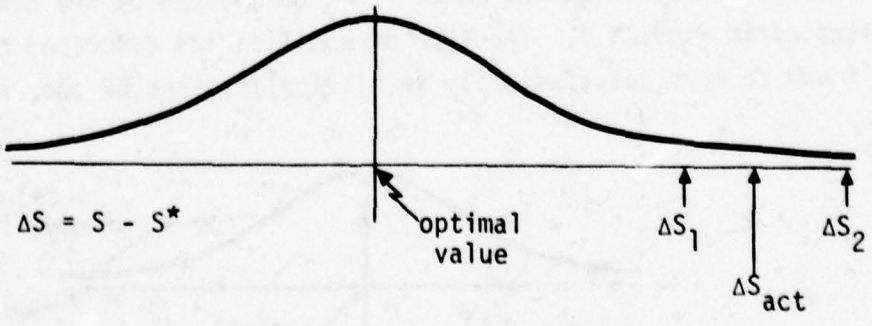


Figure IV.4. Illustration of Problem Arising in Bounded Value Functions for Large Variations from Preferred Value S*.

represent the differential between the value of the closing speed for a potential alternative and the preferred value of the closing speed. Let the scale factor S_{scale} represent the largest change in S that could reasonably be implemented during the projection period for the set of alternatives under consideration. This would correspond in practice to a major effort to reduce the closing speed of the aircraft. Since, however, in TAC BRAWLER the projection periods are typically only of 2 sec duration, the absolute change in closing speed is not significant.

Now the modification to the value be such that for

$$\frac{\Delta S_{act} - \Delta S_{alt}}{S'_{scale}} > 0$$

the value function increases, whereas for it less than zero the value function decreases. This can be achieved by writing the revised value function in the form

$$V = (1 - \gamma)C(\Delta S_{alt}, S_{scale}) + \gamma R_1(\Delta S_{act} - \Delta S_{alt}, S'_{scale})$$

where the most satisfactory value of the weighting factor γ is typically about 0.25.

The modification to the value function induced by the Reward function is illustrated in Fig. IV.5. The type of modification described here has been found to work satisfactorily in all applications of the value-driven theory.

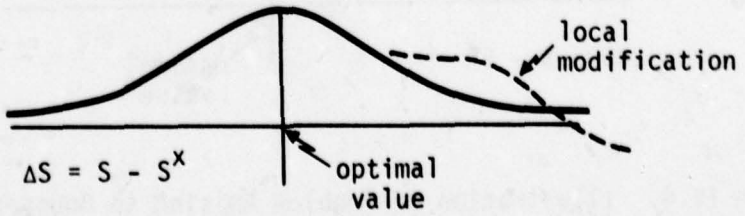


Figure IV.5. Local Modification to Value Function Induced by Addition of Type II Reward Function to Cauchy Function.

C. VALUE-MEDIATED COMMAND LANGUAGE

One of the most useful and powerful features of the value-driven decision approach is the capability to readily and efficiently communicate orders or information between different decision elements or different levels of the same element. Moreover, the form of the transmission and its effect on the receiving decision element are structured so as to encourage the decision element to act in accordance with directives but not to follow them blindly. Thus, a wingman in a flight or a company commander receiving orders from his flight leader or battalion commander to attack a hostile aircraft or to take a certain hill will generally attempt to do so, but only after considering all factors that come to bear on the decision.

The basic mechanism used to communicate information or orders in value-driven decision systems is the transmission of "revised" values for parameters in the receiving decision elements value function. The procedure is very simple and can be readily implemented in nearly all cases. We will indicate the specific form that the procedures assumes for the value function forms described in this chapter. The procedure, however, as will readily be apparent, is of much more general applicability.

To illustrate the approach consider the air combat scenario in TAC BRAWLER in which there is a flight leader, a wingman, and one hostile aircraft in the area. Among his varied responsibilities, the flight leader has the specific responsibility for "positioning" the wingman relative to his own aircraft and for committing the wingman to attack the hostile.

In accordance with the standard functional form developed earlier in the chapter, the wingman's value function has the form:

$$V = \sum_i \alpha_i v_i$$

where the α_i's are weighting factors and the v_i's are specific attributes contributing to the value of an alternative. (Recall that each alternative is described by a value function of this form. The specific value associated with the alternative depends on the desirability of the attributes for this alternative.) Now the flight leader can modify this value function in either of two ways: he can modify the weights α_i, thus changing the relative importance of the attributes; or he can modify the internal parameters, which characterize the v_i's. With these two types of modifications, the flight leader can strongly and precisely influence the behavior of his wingman.

To make the discussion explicit, consider only the attributes concerned with station-keeping, that is maintaining the correct position of the wingman relative to the flight leader, and with the initiation of an attack by the wingman on the hostile. For simplicity, assume that the station-keeping activity requires only one coordinate to specify the position of the wingman with respect to the flight leader.

Initially both the flight leader and the wingman are aware of the presence of the hostile, so that the wingman's value function includes both station-keeping and attack attributes. Its form may be represented as

$$v = \alpha_1 C(x, x_s) + \alpha_2 (v_{hostile} * P_{k_{hostile}} - v_{self} * P_{k_{self}})$$

where $C(x - x_0, x_s)$ is the Cauchy function given by

$$C(x - x_0, x_s) = \frac{1}{1 + \left(\frac{x - x_0}{x_s}\right)^2}$$

where x_0 is the desired position of the wingman relative to the flight leader, and the second term is the value of attacking the hostile aircraft, as described earlier in the chapter.

Before the initiation of the engagement the station-keeping attribute dominates the value function. This represents the nominal or default situation and is characterized by $\alpha_1 \gg \alpha_2$. In this situation alternatives for which the wingman maintains the specified distance from the flight leader are preferred over all other alternatives, including those involving the launch of an attack against the hostile aircraft. In TAC BRAWLER the default values of (α_1, α_2) are inputs to the model.

The initial step in preparing the wingman to launch an attack against the hostile is to move him to a new position from which an attack can more readily be launched. This is accomplished by changing the internal parameter x_0 from its default value to the new desired value. This makes alternatives whose station-keeping position corresponds to the new location preferred to those that do not.

To initiate the actual attack the flight leader modifies the weighting factors, so that $\alpha_1 \ll \alpha_2$. If the final factor in parentheses is positive, the attack will then be initiated. To terminate the attack, the value ordering of the weighting factors is reversed and x_0 is set to the desired station-keeping value.

In staging a real attack, a number of other considerations are made, most of which are discussed in Chapter VII. Some of these relate to the time a command, i.e., a parameter change remains in effect. In many instances it is appropriate for the command to automatically terminate at some specified time or for the parameter reflecting the command to decay to its original value. Such features can be treated by obvious extensions of the above discussion.

A few additional comments can be made concerning the term expressing the value of initiating an attack in the value function. In that expression the intrinsic value $v_{hostile}$ and v_{self} have essentially become weighting factors for the surrogate probabilities $P_{k_{hostile}}$ and $P_{k_{self}}$, which have in effect become the valuative components. The flight leader can thus modify the relative value of these components to promote an attack on a hostile. While this is of most value when multiple aircraft are involved, in which case it serves as a selection mechanism on which aircraft to attack, it is also of value in setting either a conservative or a liberal strategy for the wingman. If v_{self} is small relative to $v_{hostile}$, for example, the wingman will stage an attack without much consideration to the jeopardy in which he places himself by attacking. On the other hand, if the relative size of the value terms are interchanged, he will ascribe great value to his own safety and attack only when the situation is particularly advantageous.

By the use of such methods a model designer cannot only represent the command and information flow between decision elements but also can induce the decision entities to behave in accordance with specific strategic doctrines. This thus provides one of the most fundamental and powerful features of the value-driven approach.

V. FORMULATION OF ALTERNATIVES

The two features which are most characteristic of the value-driven approach are the explicit generation of alternative courses of actions and the association of values with the actions. These two features serve to sharply differentiate between the value-driven approach and the more commonly used decision-rule approach, in which a course of action is selected directly from the current state of the system by comparing some characteristic parameter to a series of threshold parameters. As previously mentioned, such a mechanism is commonly used in determining force postures in ground combat models by comparing ratios of firepower scores (or the equivalent) to posture threshold parameters.

In the preceding chapters the design of values to serve as the means for selecting among a candidate set of alternatives was discussed in some detail. In the examples described, the generation of alternatives did not pose any significant design problems. In the Scheduler, for example, the alternative courses of actions consisted of product subsets that were candidates for production during the next production period. In many situations, however, such as in ground combat, the enumeration and generation of potential courses of action can present significant problems. In this chapter we will examine the problems associated with the generation of alternatives and the means that have proved most successful in dealing with them. Examples will be selected primarily from ground combat applications where the most significant problems in formulating courses of action have arisen.

A. GENERAL CONSIDERATIONS IN THE FORMULATION OF ALTERNATIVES

In the standard dynamic programming problem involving at each stage a discrete set of alternative decision options, the problem of generating courses of action does not arise. The only issue is the determination of an optimal course of action. The situation is only slightly more complex in the linear programming problem in which to apply the Simplex method an

initial feasible solution must first be determined. Although slightly complicating the optimization process, the initial solution may nonetheless be determined using standard procedures. In contrast, in the ground combat problem it is not only difficult to obtain an optimal solution, but it is often difficult to enumerate fully or even to define the alternatives precisely. Penetrations, envelopments, and so forth are both difficult to define explicitly and more difficult, in fact often impossible, to fully enumerate. Moreover, the generation of the courses of action can require excessive computer time.

As a consequence of these problems, the approach that is generally taken in value-driven decision systems is to seek an "adequate" course of action rather than an optimal one. Thus, the emphasis in developing a solution shifts from one of developing high power optimization procedures for selecting the most desirable solution to one of ensuring that a sufficiently representative set of "candidate" solutions are constructed to ensure that a desirable solution is selected.

In pursuing such an approach a number of specific issues arise: How does one assure that a representative set of alternatives are examined? What are reasonable stopping rules for terminating a search for alternatives? How far into the future should alternatives be projected? What procedures are most useful in constructing alternatives? And, finally, how does one construct a hierarchy of alternatives such as is required in hierarchical decision entities? These questions are addressed in the following sections.

B. GENERATION OF REPRESENTATIVE SET OF ALTERNATIVES

The generation of a representative set of alternatives covers two different types of questions. One is strictly mathematical: given a space of feasible solutions, how does one construct a representative set of solutions? The second is more practical: how does one ensure that solutions that might be adopted by a real commander are examined?

The first of these questions--in effect, the completeness of the set of candidate solutions examined--does not generally present a serious practical problem. Either it is clear how to select the solutions or they can effectively be constructed by using very short time horizons. The latter approach permits complex solutions to be built up from short time approximations, which when taken together constitute a good approximation to a full solution. The approach essentially narrows the set of solutions requiring examination by making a series of decisions over time, each decision of which specifies the form of the solution over the time interval it covers. The problem then becomes one of establishing a heuristic value structure that is sufficiently powerful to make reasonable selections for short time horizons.

The use of the technique is effectively illustrated by TAC BRAWLER. In TAC BRAWLER the space of potentially feasible alternatives for an attacker theoretically spans all possible maneuvers in space. Moreover, those that are in any sense reasonable depend on the response of the target to the attack--a determination that obviously cannot be made accurately in advance. The direct generation and examination of all feasible maneuvers in TAC BRAWLER is thus not feasible. Rather, the procedure that is adopted in the model is to make only 2-sec time projections and then to reexamine, i.e., to make new decisions every second. The values are structured, so as to guide the aircraft toward a future projected intercept point. This solution approach is found, retrospectively, to produce reasonable trajectories for the attacking aircraft. The use of this type of approach, which depends heavily on the ability to develop effective short-term value functions, is found to work successfully in a wide variety of situations. Further discussions of the impact of time horizons on alternative generations is presented in a later section of this chapter.

The second question--the assurance that alternatives selected are representative of those that a commander would normally consider--is by far the more interesting and important question, for one generally wants

to ensure that the simulation behaves in accordance with established doctrine and tactics. The failure to act in accordance with such policy has constituted one, if not the major, criticism of theater-level combat simulations.

The most general approach that might first appear plausible for treating the problem is to allow the simulation, that is the decision elements, to consider a large number of possible courses of action and then to use the value structure to select from among them one that is consistent with established doctrine and tactics. For a number of reasons this approach is generally not desirable. First of all, it conflicts with a very general design principle which in effect states that, since simulations in general tend to become complex, it is desirable to opt for simplicity whenever possible. This applies in the design and the consideration of multiple alternatives. Thus, in nearly all cases, it is strongly undesirable to provide excessive freedom in the number of alternatives examined. More fundamentally, however, it is very difficult to build directly into the value function all significant features that contribute to a real-world decision policy. Moreover, the essential considerations tend to differ for opposing sides in a conflict.

To make the argument explicit, consider the situation in Central Europe, in which the Warsaw Pact doctrine for attacking off the march, advancing from 30 to 100 km a day, and for night fighting are strongly at odds with more conventional NATO tactics. To expect a simulation to automatically select these tactics out of a much more general set of candidate alternatives is expecting too much of the simulation if for no other reason than because details of the force structure and training of the forces are omitted from the model. In Western Europe, for example, a river that must be crossed is encountered on an average of about every 30 km. To cross these rivers and to achieve the prescribed advancement of 30-100 km a day, the Soviets have developed an extensive bridge building capability that would permit the necessary bridges to be built overnight.

Similarly, a large traffic control force has been formed to control the movement of forces attacking off the march. Finally, the Warsaw Pact forces are trained to fight in accordance with the prescribed tactics. Thus, any realistic representation of combat should reflect these actual tactics.

The most promising way that has been found for dealing with the representation of tactics is to design the simulation so that it considers only tactics that are consistent with the established tactics and then to use the value structure for selecting specific courses of action that are consistent with the general tactics for implementation in a given situation. Some variation on this scheme is feasible by slightly enlarging the tactical set and then by varying the parameters of the value structure to influence the character of the specific tactics adopted. One might wish, for example, to influence the relative aggressiveness of the forces. This could readily be accomplished by such parametric variation of the characterizing parameters.

In summary, in the value-driven paradigm the approaches that are most commonly followed for limiting the number of alternatives examined are of two types: (1) the use of short time horizons to limit the exorbitant number of alternatives that grow out of each initial alternative and which would require detailed consideration were large time horizons used, and (2) the use of doctrine and tactics as a means for limiting the number of alternatives requiring examination and, moreover, for ensuring that realistic courses of action are selected.

C. ALTERNATIVE GENERATION PROCEDURES AND STOPPING RULES

An important consideration in the design of a value-driven decision system is the determination of procedures for generating the set of alternatives that are considered as candidates for implementation by a decision element. Two general procedures are commonly employed: in the first, a complete set of alternatives is generated and then it is determined which

alternative is optimal. This procedure corresponds to the approach commonly used in format optimization theories. In dynamic programming, for example, one has a complete specification of the set of alternatives; the problem is to find the alternative that is optimal.

In the second procedure, alternatives are generated and evaluated one at a time until an acceptable alternative is found. Such procedures are commonly used in situations in which alternatives are not easily specified or generated. As described in the last subsection, this is a situation that particularly arises in ground combat applications; penetrations and envelopments belonging to this classification. Such procedures are commonly used in situations in which the number of alternatives or the dimensionality of the space is too large to allow methods such as dynamic program or gradient techniques to be readily applicable, in which case the second procedure is often computationally more efficient.

The second procedure has a number of other desirable features. In many situations the designer of a value-driven decision system is interested in representing the decision-making processes of the human decision-maker. Studies of many decision-makers in combat situations has revealed that the second procedure corresponds more closely to the decision-making procedures used by most decision-makers. A decision-maker will typically consider one or at most a few possible solutions to a problem and will then only examine additional solutions if he does not find one that is acceptable to him. This is exactly the method of the second procedure.

By adopting the second solution method, it is thus also easier to reflect the cybernetic limitations of the human decision-maker. The human decision-maker does not generally have the time or the cybernetic resources to examine all possible solutions. Moreover, even if he were aware of all solutions, he would probably be unable to select the best one--since he could not project all consequences of a solution--but could probably select a good one. The second procedure makes this process particularly easy to model.

Finally, the second procedure has the advantage of simplicity; no complex or involved mathematical tools are required to apply it. This is a particularly desirable feature in complex simulations, for if it is not adhered to, the complexity of the simulation could easily get out of hand. Thus, whenever it is feasible and whenever it provides a sufficiently accurate description of the system to be modeled, the second procedure is used in the design of value-driven decision systems.

Implementation methods for the first procedure are no different in value-driven systems than in other systems. Therefore, we will not provide any detailed discussion of implementation procedure for it. For the second procedure, however, a special problem arises--namely, how does the decision element decide if it has found an adequate solution? Or more generally, what constitutes adequate stopping rules for value-driven systems?

For systems in which it is at least formally possible to specify alternative solutions, so that Generalized Lagrange Multiplier theory can be used, but for which it is not possible or convenient to determine an optimal solution--because appropriate Lagrange multipliers cannot be found because the solutions space is not convex--an upper bound on the optimal solution can nevertheless be established.* This then serves as a basis for assessing the desirability of a candidate solution or, more formally, for establishing stopping rules.

To see briefly how this occurs, recall that in Generalized Lagrange Multiplier theory, the solution procedure consists of first selecting a set of unconstrained multipliers, performing an unconstrained maximization of the Lagrangian, and then, in retrospect, determining which constraints

*Typically, one can specify the feasible solution space, since all the space is defined by a set of constraints. Explicit enumeration of all feasible solutions, even in the case where the space is discrete and finite, is, however, generally very difficult.

are satisfied. For constraints that are violated, e.g., because too many weapons of a particular type are employed, the Lagrange Multiplier corresponding to the constraint is increased--the multiplier representing the effective cost of the resource--and the unconstrained optimization is then repeated. The effect of increasing the cost of the resource is that less of the resource will be used, thereby increasing the likelihood that the constraint will be satisfied in the reoptimization. For resources that are underused exactly the opposite procedure is followed. The repetition of this process through multiple cycles will generally--but not always--lead to the optimal solution.

The significant point associated with the optimization procedure is that for each unconstrained optimization an upper bound on the optimal solution for the desired constraint problem can be obtained, even though the unconstrained solutions do not satisfy the required constraints. In effect, the Lagrangian multipliers represent directions numbers for a plane tangent to the solution space at the point of the unconstrained solution. This plane "overlies" the solution space and thus provides an upper bound on the optimal solution. One can then use the upper bound as a means for generating stopping rules for terminating the search. Generally this takes the form of setting threshold levels that are some fixed percentage of the upper bound.

Unfortunately, although this approach is applicable to a large class of practical problems, there are many problems--namely, those for which all alternatives cannot be generated or enumerated--for which the approach cannot be used. In these cases three other methods can be used to generate stopping rules:

1. Doctrine Controlled. In this procedure the acceptability of a solution is related to strategic or tactical objectives. A simple example relates to the situation referred to earlier in which the Warsaw Pact doctrine is to advance 30-100 km per day. In this situation the value

function, as specified earlier, is given by

$$v = \left(\frac{\ell}{100}\right)^2$$

One can use this criterion as a means for establishing a threshold; e.g., to be accepted as a solution for implementation, the value of the solution must satisfy the relationship

$$v \geq 0.75$$

Variations on this approach form the most common mechanism for establishing stopping criterion in the value-driven approach.

2. Percent Improvement. This is in some sense a variation on the last procedure. The procedure consists of continuing to explore alternatives until the percent improvement in the value of the alternative falls below a specified cutoff. Great care must be exercised in applying the procedure, for if a set of closely related alternatives are examined sequentially, the improvement from alternative to alternative may be small even though one is not close to an optimal solution. The safest way to apply the procedure is to consider several series of solutions, each of which differs substantially from the other series, find the "local optimum" for each series, and then to apply the optimization criterion to the new series of optimal solutions. In this way the danger of selecting a poor solution is minimized.

3. Rigid Cutoff. A rigid cutoff provides the mechanism by which a search is terminated if other stopping rules are not effective. It is always required in a simulation to ensure that the search time does not become excessive. When a cutoff has been reached, the usual procedure is to accept the best solution that has been obtained prior to reaching the cutoff.

The rigid cutoff procedure is particularly useful in situations for which several types of alternatives are under consideration. For example, a decision element might consider the possibility of initiating a penetration vis-à-vis maintaining his current position. As a first step, the decision element might generate several candidate penetrations to determine the overall feasibility of initiating a penetration. If an attractive alternative is found, the decision element may continue to refine the penetration; if not, the rigid cutoff will terminate the search, and the decision element will consider other types of possible action.

In summary, in generating alternatives a designer has the option of either generating all alternatives initially and then seeking the optimal alternative or generating alternatives one at a time until an acceptable alternative is found. The latter procedure is generally best used when the alternatives are difficult to generate or when one wishes to simulate the human decision process. In this case the principal problem for the designer is the definition of appropriate rules for terminating the search. In most cases these must be determined heuristically.

D. PROJECTION HORIZONS

One of the more significant issues in the development of a value-driven decision system is the determination of how far into the future to project each of the alternatives generated by a decision element. Numerous factors come into play in the determination, some of which are concerned with the proper representation of the human decision-maker and others which are concerned with computer efficiency. In this subsection these issues are reviewed and their importance clarified.

In Chapter I, Section D, the general character of variable time horizons was examined. It was shown there that in the zero time projection limit the value-driven method approaches the classical decision-rule method--with the exception that multiple alternatives rather than a single, predetermined course of action is considered--and that in the long

time limit the method tends toward the classical optimization (game) problem. It was furthermore shown that each of these limiting cases imposed constraints on the development of a simulation that for many applications of practical interest make them undesirable. For example, the zero time projection, in many instances, places too great a burden on the designer to develop decision mechanisms that can adequately select desirable alternatives based solely on the current state of the system. Moreover, it provides no capability for constructing penetrations or other movements that are critical to ground penetration problems and that are so peculiar to the particular situation that they cannot be developed in advance. On the other hand, long time projections generally lead to intractible mathematical complexities and tend to produce a degree of insight that cannot be realized by a real commander. A simulation employing such long time projections, therefore, tends to to reflect real-world behavior.

For these types of reasons value-driven decision systems have tended to be designed with intermediate time horizons long enough to permit alternatives to be constructed and evaluated but short enough to avoid the mathematical complexities and lack of realism that come with long time horizons.

Two types of considerations have generally been employed in value-driven decision systems as guidelines for selecting suitable intermediate time horizons. These are: (1) the accurate representation of the capabilities of the human decision-maker, and (2) the efficient utilization of available computer resources. To employ these guidelines, a number of factors need to be considered:

1. Mathematical Complexity. In projecting alternatives the primary mathematical concern is the extensive branching of an initial candidate solution into alternative branches that strictly speaking must be individually analyzed to evaluate the initial solution. Formally, this

process gives rise to the classical decision tree of Analytical Decision Theory. In developing combat simulations it is seldom appropriate to model the decision process in such detail. A much simpler representation is nearly always preferable. In the last section it was pointed out in the case of TAC BRAWLER that the decision tree could be avoided--and at the same time the full solution constructed with considerable accuracy--by selecting very short time horizons and employing correspondingly short decision intervals.

An alternative approach, which is in many situations more effective, is to perform the projection of alternatives at a higher level of aggregation than is used for the detailed play of the simulation. This is most clearly illustrated in the Merck Scheduler. In the Scheduler, decisions to put new products in production or take old ones out are made weekly. Thus, if detailed projections were made only a few weeks ahead, the number of alternatives into which an initial alternative could branch would be enormous. Rather than proceed with such detailed projections, the Scheduler assumes that each product will run for its normal run time--or more precisely, until it reaches it maximum normal inventory--and then makes its evaluation on this basis. By using this aggregated, approximate projection procedure, the Scheduler has sufficient information available to make a determination of an appropriate product set to put into production.

By the use of such aggregation procedures projection horizons can be extended considerably beyond what would be practical were detailed projections made. For most situations the increased information accrued by using the longer horizon more than compensates for the loss of information incurred by not carrying out the detailed projections. The procedure is especially attractive for combat simulation in that the decision element ordinarily does not have available the information necessary to do the detailed projection. The aggregation procedure is thus very general and can be extensively exploited in the development of combat simulations.

2. Uncertainty. Uncertainties in the projection of alternative courses of actions, derived either from the actions of a rational opponent or from inherent stochastic variability, is a feature that tends to favor short time projections and which is closely related to the mathematical complexity issues described above. For stochastic variability, for example, the number of possible alternatives that evolves from each "chance" event is typically extremely large and one typically has no real hope of ever evaluating all alternatives. A formal treatment in this case requires the association of values with each terminal branch of the decision tree, the assignment of probabilities with each of the branches radiating from each node, and the subsequent computation of the expected payoff at each of the nodes. An optimization such as the classical "average-out fold-back procedure" of applied Decision Theory, then must be used to determine the preferred action. For most combat simulations the inherent complexity of this process makes this approach infeasible. One of the preceding techniques must be used.

When, in developing a simulation, an attempt is made to faithfully represent the human decision-maker, the treatment of uncertainty becomes a much more manageable problem. A decision-maker in a higher uncertainty environment will typically concentrate on gathering information and will only project alternatives for very short distances into the future. This shortening of the projection horizon is illustrated in a game called kriegspiel. Kriegspiel is a variation of chess in which the players do not see the opponent's forces but instead receive information from an umpire on the legality of their moves, on whether a piece is captured, and on whether their opponents or themselves are in check. Observing a game of kriegspiel, one quickly notes that the players devote virtually all of their cybernetic resources to determining the positions of the opponent's pieces. Very little attention is given to projection of future projections; the emphasis is almost completely on the gathering of information. To the extent this situation is duplicated in combat, as it often appears to be, the modeling task of the simulation designer is simplified--at least from the perspective of the complications that arise in using extended projection horizons.

3. Human Characteristics Factors. As we have repeatedly mentioned, the desire to faithfully reflect the limitations on the human decision-maker's decision process favors the selection of reasonably short projection times. Long projection times are not only computationally infeasible for most real-world systems that one would like to adequately model, but also wholly fail to capture the limitations of the human decision-maker--the projection to the end of the game representing the extreme example. On the other hand, for somewhat more subtle reasons, the selection of projection times that might at first sight appear to be reasonably representative of human decision processes can also lead to unsatisfactory results. The reason is simply that the human decision-maker, making use of his experience and of information not specifically represented in the combat simulation, can make appraisals of alternative courses of action that would appear to require a substantially greater look-ahead capability than he actually has at his disposal. For this reason, somewhat longer look-ahead times than might otherwise seem appropriate are sometimes found to provide the most satisfactory representation of the human decision-maker. The precise look-ahead time which is most satisfactory depends on the level of sophistication of the value function, which we discussed in some detail in Chapter III and will mention again below.

4. Other Considerations. The selection of a suitable projection horizon depends not only on the factors cited above but also on the ability of the simulation designer to develop adequate selection criteria for progressively shorter time horizons. For, if a value function could be adequately constructed without any projection, it would nearly always--from the perspective of simplicity of design and program running time--constitute the most desirable alternative. Projections are to a considerable extent made to simplify the problem of value design.

In fact, the simulation designer generally has a range in which he can select an appropriate time horizon and still meet all of the previously discussed objectives. The decision as to the most suitable value represents a trade-off between the effort required of the designer to develop the

value structure and the complexity and running time of the simulation. For, in the former case, the model designer, before building the model, examines the consequences of pursuing alternative courses of action, relates them to the initial state, and then by building his judgments into the model, ensures that the model dynamically selects desirable alternatives. In the latter case, the procedure is still followed, but now the emphasis falls on the model itself to generate alternative courses of action, to associate value with the outcomes, and to select the preferred actions.

In certain problems, the nature of the process for generating alternatives--such as penetrations for ground combat--require that the projection horizons be longer than would otherwise be required. Such problems came under the heading of special generation problems and are discussed in the next section.

E. SPECIAL GENERATION PROBLEMS

For most applications in which it is possible to use relatively short time horizons, the generation of alternatives is not difficult. This is true even for situations such as are modeled in TAC BRAWLER in which the number and ultimate complexity of the alternatives is substantial. By restricting the decision element in TAC BRAWLER to 2-sec time horizons and 1-sec time intervals between decisions, the generation and selection of complex alternatives can be effectively reduced to a sequence of decisions and series of individual solution elements, each of which approximate a portion of a complex solution and which taken as a group constitute a complete solution. As previously indicated, this procedure represents a very general technique that can be applied in many applications of the value-driven decision approach.

Problems in generating alternatives arise in situations in which solutions cannot be generated one piece at any time but rather the entire--or at least a major part of--the solution must be generated before it can be evaluated. Such problems are associated with the non-convexity of the solution generation space. In the development of the value-driven

approach they were first encountered in conjunction with the subtractive ABM defense problem. In that problem a set of terminal defense missiles are assigned to defend a specific target. Attacking ICBMs must then destroy all of the terminal defense missiles before the defended target can be attacked and therefore before any damage can be done to the target. The standard allocation procedure, in which ICBMs are assigned one by one to the members of a target set according to the target which yields the highest marginal payoff, does not provide an optimal solution in this case, because the marginal payoff at each defended target before the terminal missiles are completely destroyed is zero. Undefended targets will thus always be attacked before defended ones, even though if several ICBM missiles were assigned to the defended target the resultant damage would be greater than the cumulative damage from the attacks against the undefended targets. In attacking such a problem one must look ahead to the "long time" consequences of making an allocation decision. The approach used in TAC BRAWLER therefore is not applicable.

The non-convexity problem is of special interest in ground combat applications in which such activities as penetrations and encirclements play a critical and often a decisive role. For example, a penetration, judged solely by its initial effect on the relative combat advantage of the opposing forces, nearly always constitutes an undesirable action, for it (momentarily at least) weakens the defensive posture of the attacker. The complete penetration must be generated before its utility can be adequately assessed.

Special methods have been developed in the value-driven approach for generating non-convex alternatives. These involve the use of values and value enhancement procedures. We illustrate the approach by a somewhat simplified description of how penetrations and envelopments could be generated for Warsaw Pact forces in Central Europe.

The Soviets have a stated objective of advancing 30-100 km per day in a confrontation in Western Europe. This suggests that for a given front (FEBA*) an appropriate value function is given by:

$$v = \left(\frac{\ell}{100}\right)^2 \qquad \text{(V.1)}$$

where ℓ represents the distance measured from the front. The quadratic form is used to encourage deep penetrations.

For this value function, the following algorithm may be used to generate a penetration:

- Identify militarily "attractive" positions. Positions are attractive either because of their defensibility and/or because they represent desirable locations for launching future advances. Such positions must typically be identified by the user as input to the simulation. Figure V.1 illustrates a representative scenario.
- Associate values with the identified positions. In the simplest case the values can be assigned directly from Eq. V.1.
- Identify possible penetration points along the front. The identification of these points typically depends on terrain characteristics and on estimates of enemy (and friendly) force densities in the area.
- Consider links connecting all combinations of initial points and militarily attractive positions (perhaps constrained by range for computer efficiency). Associate values with the links, where the values are the value of the terminal position degraded by estimated attrition in reaching the position. By this procedure multiple values are generally associated with each position.

*Forward Edge of the battle area.

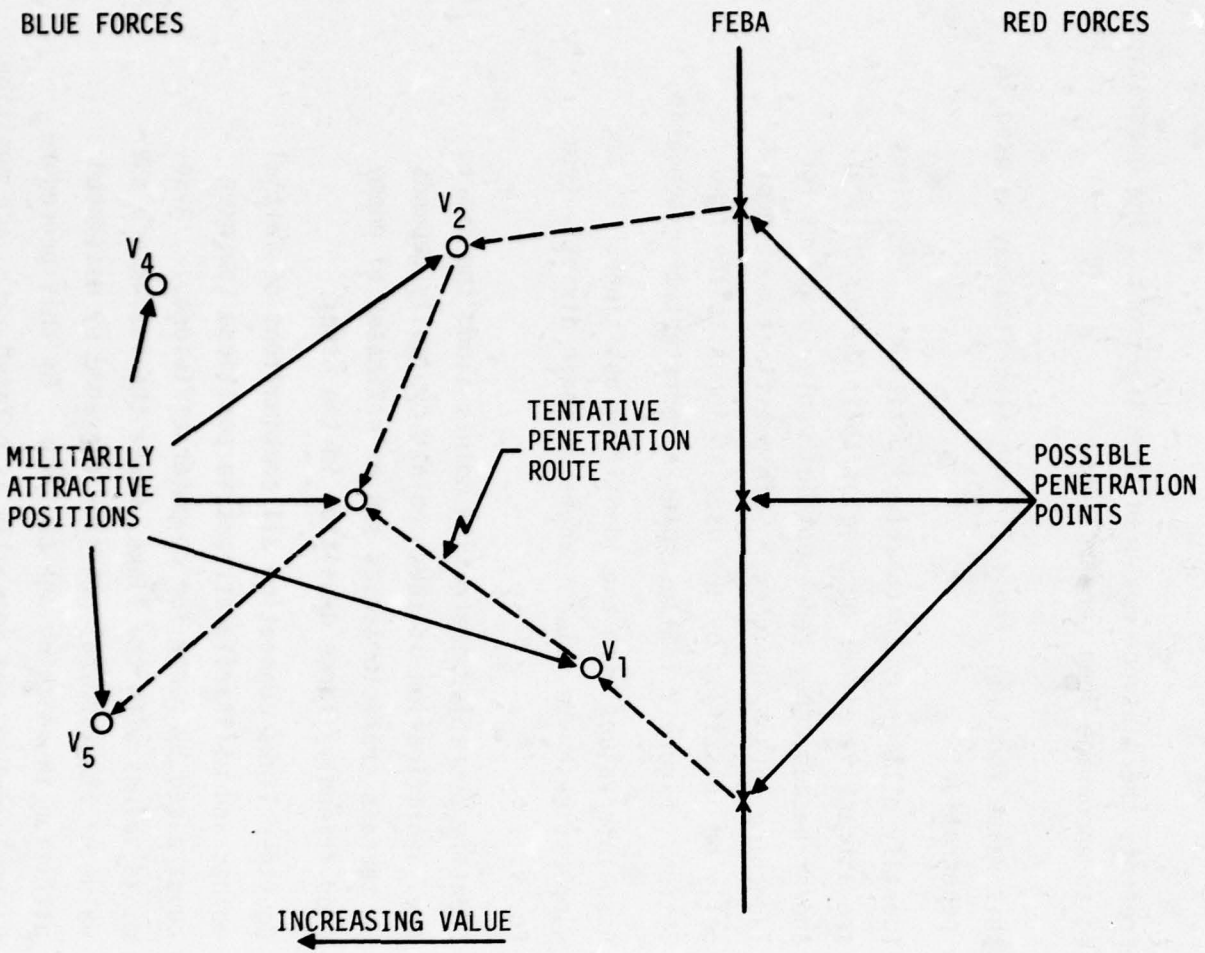


Figure V.1. Implementation of the Value-Enhancement Algorithm for Non-Convex Alternative Generation

- Determine link of highest value. Establish that link as a tentative part of the penetration structure and assign the value of the link as the new value of the position. If encirclements are desired, enhance the value of the position to encourage later links from other established routes.
- Continue the procedures by examining links from positions already reached to other positions and by assigning values to these new positions. After each link is selected for inclusion in the penetration structure, evaluate overall penetrations. Evaluation criteria should reflect the current depth of the penetration as indicated by Eq. V.1 and should account for the defensibility of the penetration and perhaps for the vulnerability of the supply lines.
- Continue the procedure until an acceptable penetration is achieved or until a fixed cutoff (perhaps in terms of number of links examined) is reached.

The algorithm outlined here provides a powerful tool for modeling penetrations. By varying the way in which the position values are enhanced, a designer is given considerable power over the way a penetration is structured. For example, a designer--or if the capability is provided as input, a user--may not wish to create encirclements until a certain depth of penetration has been achieved. This can be accomplished by enhancing only the values of positions beyond the desired depth and by diminishing values at lesser depths. Through this type of mechanism considerable control over the penetration can be realized.

It should be noted that this value-enhancement procedure is the same one that was used to encourage products to remain in production in the Merck Scheduler.

F. HIERARCHIES OF ALTERNATIVES

As the set of decision alternatives required to allow a decision element to perform in a realistic manner becomes large, it is often useful to introduce the hierarchical decision element, in which a decision element is divided into two or more functionally related parts. For example, the concept of the hierarchical decision element, introduced in Chapter I, was developed for flight leaders in TAC BRAWLER as a means of efficiently representing the different types of decisions a flight leader must make during the course of an air battle.

The use of the hierarchical decision element concept introduces a requirement for a methodology to treat alternative generation at each level of the hierarchy. In TAC BRAWLER, for example, the flight leader must function both as a flight leader and as a pilot and the decisions he makes at one level influence those he makes at the other. As a flight leader he must make decisions for the flight as a whole: aircraft allocation to targets, tactics, assurance of mission success, and the like. As a pilot he must fly his own aircraft and ensure his own personal success in an air battle. In general, therefore, the hierarchical decision element makes decisions at two or more interrelated levels. The highest level nominally performs the function of grasping the problem in broad general terms, with the lower levels responding to the decisions made at the higher levels to examine increasingly more detailed decisions. This concept is shown in Fig. V.2, for a two-tiered hierarchical decision element. Notice that there may be some overlap of action alternatives at the lower levels. Such an overlap exists, for example, when the low-level option selected is a hard pull up. Such a maneuver may have been selected from a high-level decision involving a hit-and-run or a brawl attack.

The methodology for relating decisions at different levels of a hierarchical decision element follows naturally from the structure of the element. Decisions at an upper level define a class of actions for the decision element at the lower levels. Only those actions are then considered by the lower level. For example, a flight level decision to attack

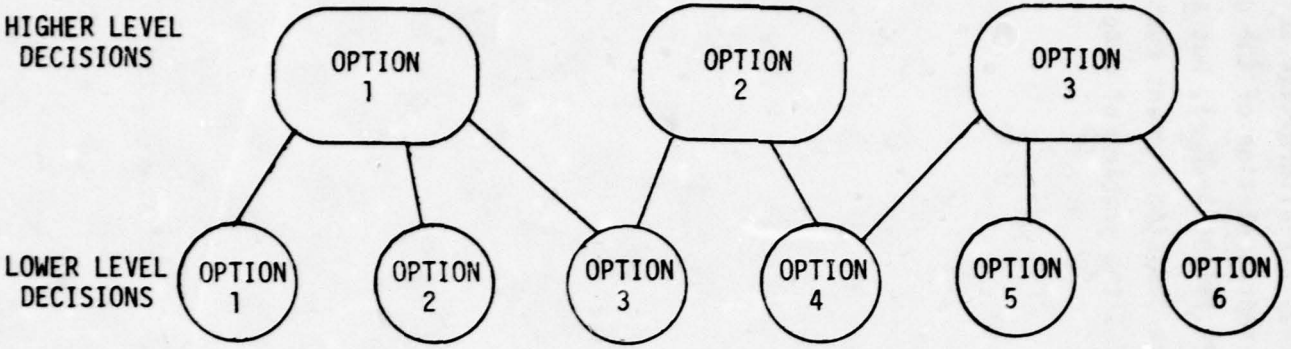


Figure V.2. Decision Alternatives in a Hierarchical Decision Element

in TAC BRAWLER constrains the pilot level to consider only alternatives consistent with an attack.

Use of the hierarchical methodology for alternative selection therefore not only allows a more realistic representation of the decision element (as in the case of the TAC BRAWLER flight leader), but also limits the number of detailed alternatives the decision element must examine. In complex systems it greatly simplifies the problem of modeling the decision process.

VI. DATA INTERPRETATION AND SITUATION PERCEPTION-- ARTIFICIAL INTELLIGENCE METHODS

In a value-driven decision system the primary function of the data interpretation and situation perception activities is to generate the parameters that are necessary for generating and evaluating alternative courses of action. In the language of Chapter I these activities are concerned with the generation of the mental model, which provides the decision element's view of the real (simulated) world and which is then used as the basis for the decision element's actions. The discussion of these two activities forms the principal subject of this chapter.

The data input activity, which is the third activity involved in the generation of the mental mode, is treated no differently in value-driven simulations than in other simulations. Typically, sensor systems--reconnaissance aircraft, FACs, space satellites, etc.--gather data which is then relayed to a command center, i.e., a decision element, for processing. As mentioned in Chapter I, uncertainties, information loss, and so forth, are generally most conveniently introduced before reaching the decision element proper. In some applications--TAC COMMANDER, for example--there is a feedback loop which directs the sensors to search a certain area. Such an action is typically initiated when preliminary intelligence indicates activity in an area. The decision to assign sensors to an area and the commitment of sensors is handled by the decision element as it would any other possible action. Thus no special features are required in value-driven decision systems for treating data inputs.

One specific procedure for generating the sensor events themselves, however, has proved particularly useful for simplifying value-driven simulations. It concerns the statistical generation of events in a situation in which the force being observed--a ground army, for example--is not explicitly modeled in the simulation. A discussion of this procedure is given in the first section. In the second section, the general characteristics of the data interpretation and situation perception

activities are described, and in the third section the use of artificial intelligence techniques as a means for implementing the activities is discussed.

A. SENSOR EVENT GENERATION

The situation is frequently encountered in combat modeling in which it is desired to generate statistically "correct" observations on a hostile force without explicitly modeling the details of the force, itself. The situation in TAC COMMANDER is typical. A simulation is needed for evaluating a tactical air command-and-control system responsible for assigning aircraft to time perishable targets detected by reconnaissance aircraft. Critical to the evaluation of the C^2 system is the proper representation of the time and space correlations among the "detection" events. For example, exceptional activity observed in an area during one day generally implies a high level of activity the following day. Activity in a rearward area one day, such as might be generated by the movement of forces toward the FEBA, implies a high likelihood of activity in the forward areas on subsequent days. Massing, which places severe strains on a C^2 system, is frequently observed. Correlations also exist among different sensor types, so that if one sensor observes significant activity in an area, other sensors should also have a high likelihood of observing significant activity. Finally, correlations should exist among different target types. The observation of Close Air Support (CAS) and Mobile targets, for example, are generally correlated.

One would like to be able to construct the set of observations without modeling the activites of the ground forces in detail. The procedure used in TAC COMMANDER for generating the observations is particularly simple and may be applied in a wide range of applications.

For illustration, tactical reconnaissance aircraft (TAR) searching an area. For a given type of target, given density of targets, a given

sweep width of the radar sensor, and a given speed of the TAR, one can calculate the mean number of detection λ per unit time. Assuming the targets are detected at random, one can simulate the time the first target's detected by sampling from an exponential:

$$f(t) = \lambda e^{-\lambda t} \qquad \text{(VI.1)}$$

The time of subsequent detections can be determined by resampling from the exponential using the time of the last detection as the new time zero. Continuing this procedure, one can obtain a set of detections during a particular time period T in a particular area for a particular target type. For the time period T, the sampling procedure produces a total number of events that is distributed according to a Poisson distribution.

This simple Monte Carlo procedure is adequate for representing detections in most simulations if correlations are not important; a separate λ and hence a separate exponential λ is defined for each sensor and target type. Problems arise only when one wishes to reflect massing and the inter-sensor, inter-time correlations described earlier.

The effects of massing and correlations among sensor events are reflected in TAC COMMANDER by correlating the λ's themselves rather than by correlating the actual observations generated by the exponential distribution and by generating the λ's from a distribution that permits large values of λ to be occasionally observed. By ensuring, for example, that λ for day two is large if λ for day one is large, one indirectly ensures that the expected number of events on day two is large if the expected number of events on day one was large. As a first step, consider the procedure for representing massing.

To reflect spacial variations in density, forward and rearward areas are divided into sections as illustrated in Fig. VI.1 with a

4	5	6
1	2	3

Figure VI.1. Typical Sector Division

separate λ corresponding to each sector. For each sector λ is generated using a Monte Carlo sampling process in the same way the actual events are generated from the exponential distribution. A normal distribution, which might seem to be the natural first choice for the sampling distribution, does not work well here, since λ nonnegative and the stochastic properties of the normal are such that massing is an infrequent event. A much better distribution is a lognormal, which is defined by the relationship

$$\mu = \ln\lambda \qquad \text{or} \qquad \lambda = e^{\mu}$$

where μ is normally distributed. λ is clearly positive and can be represented by its mean m_λ and standard deviation σ_λ in the same way as the normal. Its approximate form is illustrated in Fig. VI.2. The mean, m_λ, representing the expected number of observations observed per unit time, varies in a typical scenario from forward to rearward sectors. The standard deviation controls the intensity of the variations in λ. Along with α and w described below, it serves as a control parameter for regulating the speed of development and the intensity of the ground battle.

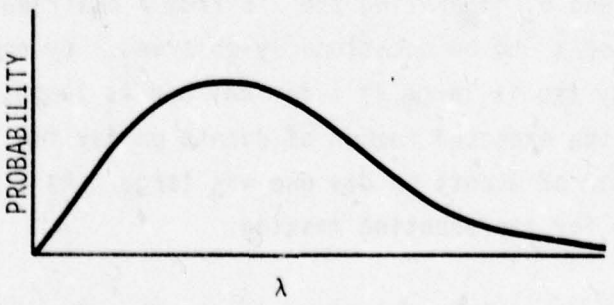


Figure VI.2. Lognormal

The effect of σ_λ and some of the desirable features of the lognormal distribution can be seen in Fig. VI.2, where for small m_λ and large σ_λ infrequent but large values of λ will arise in the sampling process. The lognormal distribution thus has the property that it will produce stochastic massing events that correspond in character to what would be observed were ground forces modeled in detail and stochastic observations made upon the forces.

Consider next the correlations over time within a specific sector. If considerable activity were present in a sector during the preceding period, on the average, considerable activity would be expected during the current period. If for each period a new sample is drawn directly from the lognormal distribution, no correlation would exist between samples; for one period the sample might have a value greatly above the mean, and for the next period greatly below. To introduce the correlations among the λ's into the model, exponential smoothing is used. This consists simply of taking a weighted average between the λ for the last period and the uncorrelated λ for the current period, according to the formula

$$\lambda_{new} = (1 - \alpha)\lambda_{current} + \alpha\, \lambda_{old}.$$

The parameter α controls the rate at which λ changes over time. For $\alpha = 0$, the current and the old λ are completely uncorrelated and change takes place at the maximum rate. For $\alpha = 1$, no weight is given to the current λ, and hence λ is constant over time. Therefore, by varying α, the speed at which the ground battle develops is varied. The speed of development puts different strains on the C^2 system.

Consider next the correlation between sectors. The inter-sector correlation is designed primarily to allow the activity in one sector during the preceding time periods to influence the behavior in another sector during the current period. A high level of activity in a rearward

sector, perhaps as realized through convoy movement, is permitted (subject to user specification) to influence the likelihood of substantial activity in the corresponding forward sector during the next period.

The event generator treats the correlation by using an exponential smoothing technique similar to the dynamic smoothing procedure used in modeling the intrasector correlation. The weighting takes the form

$$\lambda'_{new} = w\lambda_{new} + (1 - w)\left[1 + \left(\frac{\lambda - \lambda_m}{\lambda_m}\right)\right]\lambda_{new}$$

where λ_{new} is the exponentially smoothed λ for the forward sector, λ is the λ for the rearward sector during the previous period, λ_m the mean λ for the preceding period for the rearward sector, (i.e., m_λ) and w the preassigned weighting factor.

From the form of the expression, it is seen that if the rearward sector had exactly its mean level of activity during the preceding period, it has no effect on the λ'_{new} for the forward sector; if it were greater, it tends to enhance the λ'_{new}; if it were less it tends to diminish it. The expression therefore has exactly the right characteristic behavior. While the correlation is described in terms of only one rearward sector influencing a forward sector, any sector or combination of sectors may influence any other in TAC COMMANDER.

Another type of correlation must be represented between two or more sensors operating in the same sector. In general, if one sensor observes considerable activity in a sector other sensors will tend to also do so. This is treated in TAC COMMANDER by using the same random number for generating the different λ's in a sector during a given time period. Thus, while the individual λ's are sampled from the different lognormal distribution, the drawing of a large random number from the uniformly distributed distribution will tend to produce relatively large λ's for each of the sensors.

A final type of correlation allows for the presence of multiple target types. Suppose, for simplicity, there are only two types which are taken as armoured columns and troop concentrations. Ordinarily, a user will have an estimation of the composition of the enemy forces, and more specifically an idea of the relative frequency with which a particular sensor will observe each target type. Suppose that in the particular area where the TAR aircraft is operating, a detected target is an armoured column with a probability of 0.3 and a troop concentration with a probability of 0.7. To determine the target type for a particular observation, a uniformly distributed random number is drawn and compared to the probabilities. In the example if the number is less the 0.3, an armoured column is detected, if it is greater, a troop concentration is detected. This simple procedure thus provides a method for handling multiple target types.

An important feature of the method is the control parameters that can be used to regulate the development and character of the ground battle. The parameter α controls the speed of development. The w parameters control the influence of activity in one sector on another and the speed with which activity propagates. Finally, the standard deviations of the mean activities control the intensity of activity and the abruptness with which disturbances arise.

In practice, the control parameters can be used to produce a standard set of campaigns against which a C^2 system can be evaluated. The parameters or characteristics of the C^2 system are varied, evaluating each case against the standard set of campaigns.

In exercising the event generation procedure in TAC COMMANDER, events are generated for an area if and only if one or more sensors are assigned to the area. The λ's, on the other hand, reflect the state of the ground battle and are always generated in the model whether or not the corresponding sensors are present. A λ represents the mean number of

events per unit time that would be observed in a sector were a single sensor present. The actual number of events arising in an area thus depend both on the λ and on the number of sensors assigned to the area.

Accounting for the presence of multiple sensors of a particular type in an area is simply carried out by multiplying the λ in the exponential distribution of Eg. VI.1 by the appropriate factor. If there are two sensors λ is multiplied by 2. More elaborate procedures could be developed for treating multiple sensors but are seldom necessary.

The procedure discussed in this section for TAC COMMANDER is clearly widely applicable to many of the problems encountered in combat modeling. Where it is applicable, it considerably simplifies the task of modeling a combat force, so as to produce meaningful data for analyzing a C^2 system.

B. DATA INTERPRETATION AND SITUATION PERCEPTION

As mentioned in the introduction to the Chapter, the principal function of the data interpretation and situation perception activities is the determination of those parameters that are required for generating and evaluating potential courses of action. Although for many simulations there is no distinction between the two activities, for more complex simulations the two activities represent rather distinct activities. For a ground combat simulation, in which the distinction is most easily clarified, the data interpretation activity consists of the construction of a "situation map," consisting of the location--real or surmised--of friendly and hostile forces, and, where feasible, indicating their direction of movement and other characteristics that are subject to observation. The type of interpretation required in the data interpretation activity would consist, for example, of concluding from the observation of a tank, accompanied by some evidence of additional activity, the possible presence of an armoured column in an area. To draw this type of conclusion a simulation requires a considerable amount of prior knowledge. In the language of Chapter I this is contained in the Prior Knowledge Library.

The situation perception activity differs from the data interpretation activity, in that it consists not of generating the situation map but of analyzing it to determine the variables relevant to the generation of candidate courses of action. By analogy with the human decision process it may be referred to as "action-oriented perception," in that the guiding principle in situation perception is to construct that portion of the external world that is relevant to the actions that are available for implementation by the decision element. Situation perception is thus closely and intrinsically intertwined with the generation of alternative courses of action.

The primary function of the situation perception activity and, by implication, the data interpretation activity in value-driven systems is thus the generation of the parameters that are required by the decision-making process. Since the data interpretation and situation perception activities are concerned with the generation of the mental model, another way to say this is that the mental model contains that portion of the real world--or an approximation thereof--that is relevant to decision-making.

Let us next consider some examples of the data interpretation and situation perception in value-driven decision systems and how they impact the selection of courses of action. In the ground combat example cited under Special Generation Problems in Chapter IV an important factor in the generation and evaluation of a proposed penetration route was the density ρ of hostile forces in an area of proposed penetration. The determination of a precise penetration route depended directly on ρ, for ρ has used to calculate the attrition of friendly forces and the likelihood of successful penetration. It furthermore entered into the evaluation of the penetration route, particularly through its impact on attrition, which is, of course, a significant consideration in evaluating a proposed course of action.

The force density ρ is thus a parameter that enters directly into the generation and evaluation of candidate courses of action. It is a representative product of the data interpretation and situation perception activities. It may formally be thought of as having been generated from observations such as that of the tank, which through the data interpretation activity is generalized to the presence of an armoured column, and subsequently by the situation perception activity to the force density.

The force density ρ also illustrates another feature of the situation perception activity. Projections of candidate courses of actions in value-driven decision theory are generally conducted at a higher level of aggregation than is the actual simulation of the combat. This procedure has the advantage that it generally immensely shortens the running time of the simulation and makes use of information only at the level available--or surmised--by the decision element. The decision element does not know the location of every hostile--or friendly--tank and thus is incapable of making projections at the level of detail of the combat simulation. One of the most general features of the situation perception activity is thus one of aggregation. Most situation perception functions involve aggregating lower level data into a form appropriate for the decision-making function.

TAC BRAWLER illustrates a number of aspects of situation perception. In particular, it illustrates that many of the functions of the situation perception actually are concerned with assessing the intentions of hostile forces and the capabilities of one's own forces. A flight leader in TAC BRAWLER engaging a group of hostile aircraft, for example, must assess the vulnerability of his own aircraft--i.e., the flight's--to the hostile aircraft in the engagement and to the extent possible assess the intent of the hostiles, so that he can advise his flight members of the appropriate actions to initiate. This activity corresponds directly to the definition of the situation perception activity, in that it begins with the situation map--the position, heading, speed of all aircraft--and interprets the map in terms of the dangers to his aircraft and the opportunities for his aircraft to engage the hostile forces.

Once the flight leader has completed his assessment, he communicates the assessment to the flight members through the value-mediated command language. This has the effect of enhancing the flight member's value functions for those courses of action that are in accord with the flight leader's assessment. A flight member in danger of being attacked by a hostile, for example, would have his value function enhanced for those courses of action that would enhance his likelihood of evading the attack.

TAC BRAWLER also illustrates the degree to which the implementation of the situation perception activity is currently an art rather than a science. The precise method by which the nature and the degree of a threat to a flight member by hostiles is assessed must currently be developed largely by ad hoc techniques. Insight and experience in the development of simulations are the main requirements for effective modeling of such activities. The techniques of artificial intelligence, however, offer some promise for the development of more systematic implementation procedures. These are discussed in the last section of this chapter.

A final example of the situation perception activity is provided by the use of shadow values (Lagrange Multipliers) in the Merck Scheduler and the MUSTEX models. In the Scheduler, the shadow values are associated with the modules, and represent the value of adding one additional module of the specified type to the production plant. They thus represent a composite of a large amount of highly detailed information into a set of highly aggregated parameters. As described in Chapter III, they enter directly into the generation and evaluation of potential courses of action. Each shadow value may be interpreted as the rent that must be paid on a module if a product is to be put into production. The module rent must then be compared with the profit that is realized if the product is put in production to decide if it should be produced during the next time period. The shadow values thus enter directly into the evaluation of courses of action and their generation constitutes one of the major functions of the situation perception activity in value-driven systems.

The use of the shadow values in MUSTEX is similar to that in the Scheduler with the exception that in MUSTEX the shadow values are used directly to evaluate courses of action in intermediate stages. In MUSTEX, then, the situation perception activity assumes in addition to its normal situation perception function, the value generation function, for generating and evaluating the assignment of missiles to military targets.

It should be noted that the situation perception activity is also required in the decision-rule approach. In that approach, the potential combat states of the simulation are divided into a set of categories, each of which corresponds to a prescribed course of action for the forces. The situation perception function is used to determine in which of the categories the current combat state of the simulation falls. For example, in many ground combat simulations the relevant variable is the ratio of the firepower scores of the opposing forces. Once this ratio is determined, it is compared to a series of thresholds, which determine the category into which the relative strength of the forces fall, and then to the courses of action for the two forces--for example, attack for Red, defend for Blue. The process of determining the category in which the combat falls constitutes the situation perception activity in decision-rule systems.

As mentioned above, the techniques of artificial intelligence offer some possibilities for formalizing the data interpretation and situation perception functions. This possibility is explored in the following section.

C. APPLICATIONS OF ARTIFICIAL INTELLIGENCE METHODS[1]

In Section F of Chapter I Dr. Michael Arbib's characterization of the decision-making paradigm of Stanford's mobile robot was analyzed and

[1] Some of the factual material in this section was adapted from a paper, The Relevance of Problem Solving Methods in Artificial Intelligence to the Modelling of Command and Control, presented by Dr. D. Cooper at GRC to the 80th Military Operations Research Society Symposium in Monterey, California, December 1977.

compared with the corresponding paradigm of the value-driven decision approach. Many similarities between the two approaches were modeled. In addition to such characterization of the decision-making process, artificial intelligence offers many techniques for the storage, retrieval, organization, and other manipulation of data. These techniques are potentially complementary to the value-driven approach, in that they offer the possibility of developing systematic procedures for implementing the data interpretation and situation perception activities. This possibility is explored in this section.

A very basic area in which artificial intelligence techniques are potentially useful is in the formal, systematic representation of a scene such as the state of the battlefield. It would be extremely useful if the principal features and particularly the interactive features of the battlefield could be represented in a form that would simplify and systematize further analyses. Such a possibility is offered in the field of artificial intelligence known as scene analysis and, in particular, by the use of directed relational graphs. Scene analysis is the brand of artificial intelligence concerned with extracting useful information from pictures and preparing it in a form that can be used for further analysis. It differs from pattern recognition in that its primary emphasis is description rather than identification or classification.

Directed relational graphs provide a means by which a scene is replaced by a network, the nodes of which represent the pertinent entities in a scene and the links or arcs connecting the nodes contain relations between the nodes. Figure VI.3 represents a simplified Tactical Air Engagement for TAC BRAWLER and Fig. VI.4 represents the corresponding directed relational graph. The direction of the arcs are chosen to represent the direction in which each attribute acts. For example, the Red wingman is under attack by the Blue wingman, who in turn is visible to the approach Red flight leader. The "Part Of" labels on the arcs are incorporated to allow access within the data base for "higher" to "lower" level entities and, if desired, from lower to higher.

Figure VI.3. A Simplified Tactical Air Engagement for TAC BRAWLER.

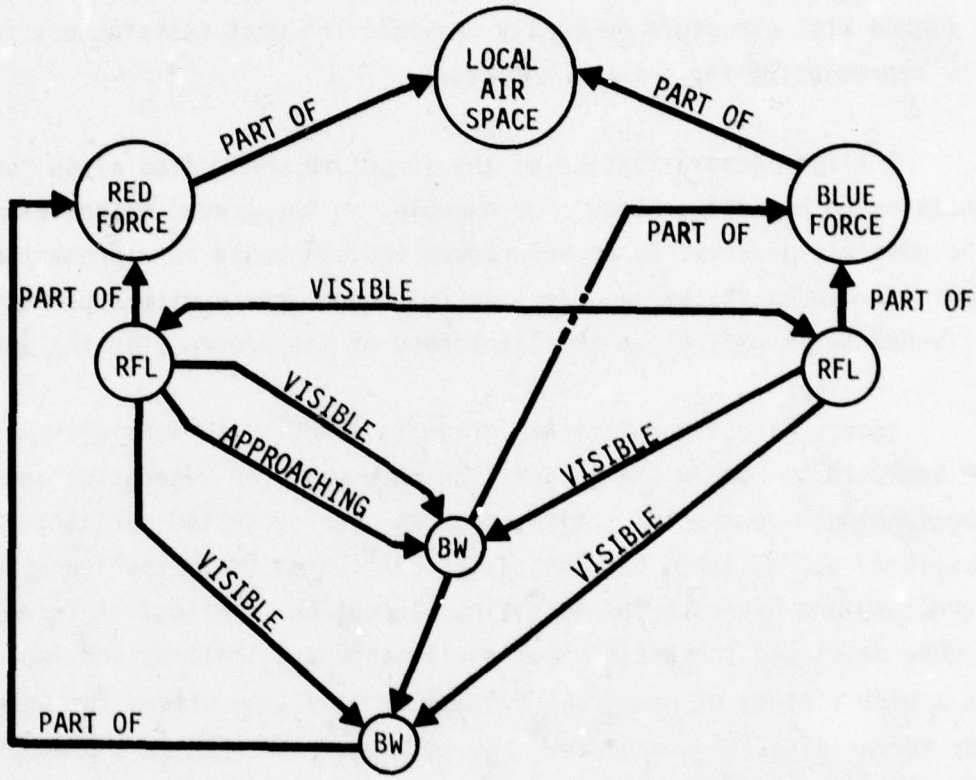


Figure VI.4. Directed Relational Graph for Tactical Air Engagement

Computer implementation of directed relational graphs consists of first defining a list of the possible attributes that are permitted to connect entities of the various types and then to test whether or not the conditions necessary for each attribute to be satisfied are met. If the conditions are met, the link is constructed. Uncertainties in the "validity" of the link can be incorporated by including a probability with the link. A linked list structure generally provides the most satisfactory technique for representing the network structure.

A slight generalization of the structure would also allow "uncertain" entities to be represented. For example, in the ground combat examples, the possible observation of an armoured column could be represented by adding a new entity to the directed relational graph with a probability attached to it reflecting the likelihood of its presence in the area.

Once a directed relational graph is constructed--the elements must be analyzed to obtain the information necessary for generating and analyzing the candidate courses of action. As has been suggested earlier, this is generally accomplished heuristically; one learns from experience how to structure and exercise the situation perception function. A formal procedure developed for artificial intelligence applications and implemented in a wide variety of practical systems of many type offers the possibility for formalizing the procedure. The technique is known as a production system or less formally as production rules.

The production system is particularly attractive because it provides a procedure for formalizing the Prior Knowledge Library described in Chapter I. Recall that that library contained all of the prior information that was required for generating and processing the mental model. This information is particularly relevant for the data interpretation and situation perception activities where, for example, without the prior knowledge that the observation of two or three tanks in a particular environment indicates the possible presence of an armoured column, such a deduction could not be made.

The prior knowledge required for such deductions is in most value-driven simulations not stored in one place; it is distributed throughout the simulation. The production system offers the possibility of condensing much of the information into a single physical library and doing so in a form such that it is readily constructed and modified by the user.

A production system may be viewed as consisting of three basic components: a set of rules, a data base, and an interpreter for the rules. The rules define what action is to be taken or what conclusions are to be reached if a given condition exists in the data base. The interpreter, which is part of the Executive Control Program in value-driven theory, controls the order in which the rules are scanned and processed.

To illustrate the use of production rules consider the following simple example of a single production rule:

(2 tanks observed by reconnaissance, moderate cover) ⟶ (presence of armoured column with probability of 0.6)

This simple production rule allows the data interpretation activity required for analyzing the presence of two tanks to be performed simply and efficiently. The simulation can be easily structured so that the precise form of the production rule, its condition, and its conclusions, can be readily specified by the user.

When a complete set of production rules is specified, the simplest procedure is to have the interpretor, either periodically or at times controlled by the occurrence of certain key events, scan the list of production rules and, for those for which the conditions on the left hold, draw the conclusions on the right, or, if an action is specified, take the specified action. The action may require another rule or series of rules to be tested. In this way hierarchies of rules can be constructed and several different conditions can be combined to draw a more global conclusion. (The possible presence of a tank column, the observation of troop movement, and an indication of a supply buildup may suggest an impending attack.)

The production rule approach can be used to specify alternative courses of action for given combat conditions. In this case the production-rule approach reduces to the decision-rule approach discussed in Chapter I, in which case it is subject to all the strengths and weaknesses of that approach. A more useful way for exploiting production in the decision process itself appears to be determining the courses of action that require examination. For example, in a ground combat simulation, the determination of whether or not to explore the possibility of initiating a penetration can be treated by the use of production rules, by considering, for example, the ratio of fire power scores of the two forces. A two-stage decision to explore the possibility of a penetration is treated by the decision-rule approach and the actual construction and evaluation of a candidate penetration by the value-driven approach.

The production rule approach is thus seen to be potentially a powerful tool for implementing value-driven decision theory in combat simulations. It warrants much more extensive examination.

Another approach that has some potential application in combat simulations, although in our judgment much less than the production rule approach, is the signature table. The signature table has been extensively exploited in the development of computerized chess, where Samuels and others have refined the technique to the point where it can play championship chess.

The signature table consists basically of a hierarchy of tables each of which takes as input a specified set of variables that assume a finite set of values and produces as output a single variable that also assumes a finite set of values and which in some sense is an aggregation of the input variables. The correspondence between the input variables and the more aggregated output variable is specified by the table entries. A representative table, constructed by Dr. Dennis Cooper of General Research Corporation (GRC), is shown in Table VI.5.

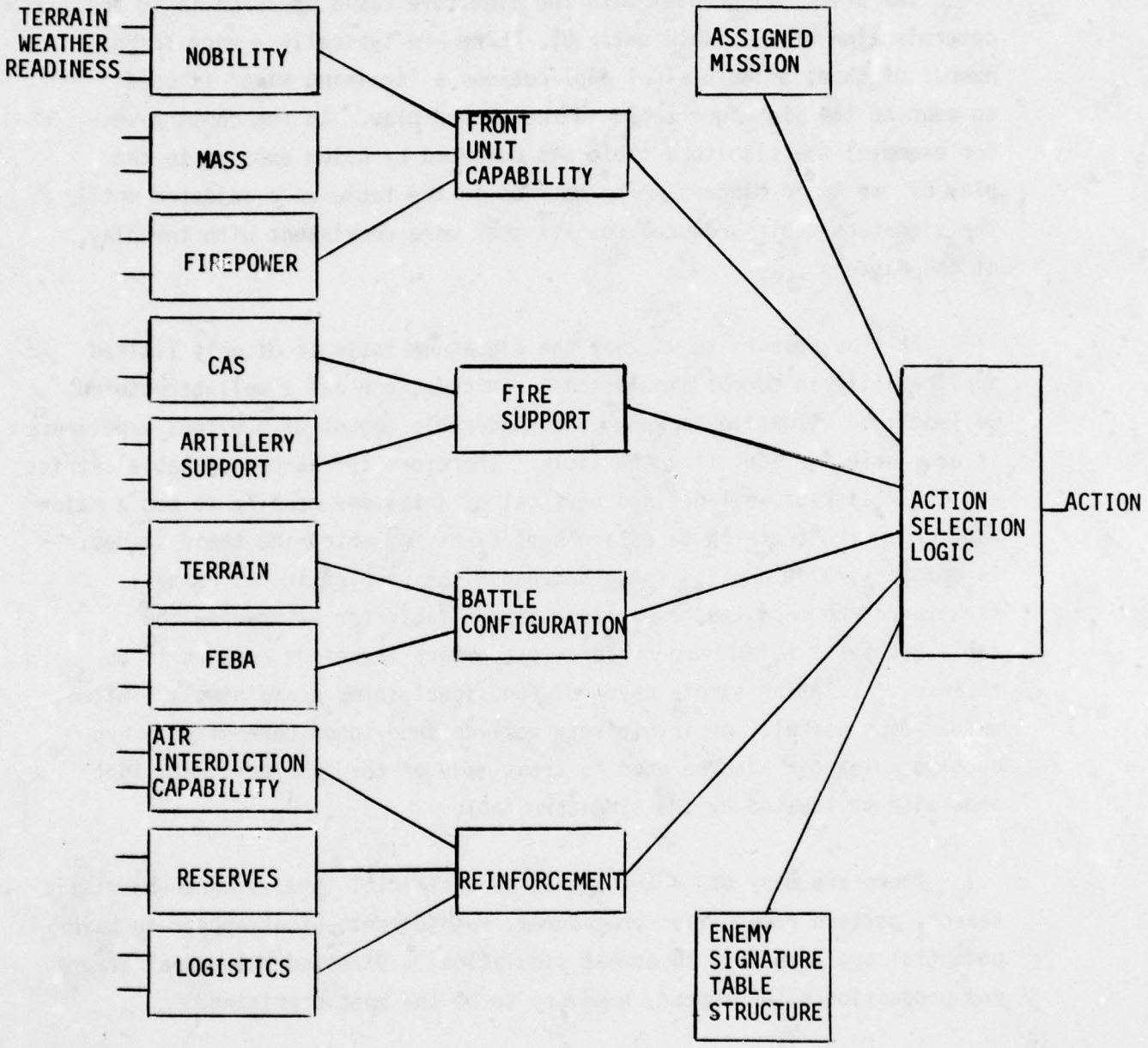


Figure VI.5. Signature Table (Adapted from Dennis Cooper of General Research Corporation)

The principal problem with the signature table approach is in the determination of the table entries. There are typically a very large number of them. In classical applications a "training tape" is used to educate the signature table to the proper play. In the chess case, for example, the signature table was educated by being exposed to the play of the Grand Masters. The entries in the table were adjusted until the signature table produced results that were consistent with the play of the Masters.

It thus appears to us that the signature table is of only limited applicability in combat simulations. In chess one has a well-structured, well-defined situation in which a considerable amount of previous experience is available for educating the table. Therefore the number of table entries and their lack of well-defined physical or intuitive meaning is not a major handicap. The training tapes are sufficient for obtaining their values. In combat simulations, on the other hand, the problem is not as well structured and much less experience is available for estimating the table entries. A far more preferable approach therefore appears to be the use of relating simple physical functional forms containing a limited number of physically or intuitively well-defined input parameters. Production rules can also be used to treat many of the problems that might otherwise be treated by the signature table.

There are many other techniques of artificial intelligence--heuristic search, pattern recognition procedures, and so forth--that appear to have potential applicability to combat simulations. Directed relational graphs and production rules appear, however, to be the most promising.

VII. COMPUTER IMPLEMENTATION

A. INTRODUCTION

The previous chapters have discussed the basic theory and mathematical development of value-driven decision theory. As a result of experience in implementing the theory in the simulations discussed in those chapters, design concepts have emerged that form the basic principles for the design and development of value-driven computer simulations.

In Chapter I, Section D, an introductory discussion of the logical structure of value-driven decision theory in TAC COMMANDER was presented. This chapter expands upon that discussion by presenting a more detailed discussion of procedures for simulating the decision-maker and for linking decision entities. In order to provide examples of these concepts, this chapter draws heavily upon the design principles developed for the most advanced of the value-driven simulations--TAC BRAWLER.

Since individual applications of the value-driven concept differ greatly in scope and objectives, this section is not intended to be a cookbook for the computer implementation of the value-driven principle. Rather, the emphasis is on general techniques and guidelines to produce effective simulations using value-driven technology.

B. SIMULATING THE DECISION-MAKER

In a traditional event-oriented combat simulation most of the information that must be stored falls into two categories: the data concerning future events, which is maintained in a time-ordered event file (list memory), and data concerning the physical status of forces, which is maintained in central status arrays. In the value-driven simulation a large amount of additional data is also required, which is concerned with the status of the mental model for each individual decision element: its perception of the environment, the activity in which it is engaged, its value priorities,

and the like. If this information were stored in the central status arrays, it would impose infeasible requirements for data storage. This problem has dictated a new design concept termed the "consciousness event."

In order to restrict a decision element to make decisions based only on the knowledge it legitimately has available to it, all decision processing is allowed to operate only on the version of the system state contained in the individual decision element's mental model. For the purposes of this discussion, it is convenient to include in the mental model both the decision element's perception of the state of the simulation (the "Perception Array") and its Prior Knowledge Library. The Perception Array differs from the Prior Knowledge Library in that it is being continuously updated based on new information from sensor and communication links. The Consciousness Event then is the vehicle through which incoming information is processed, and in which the Perception Array is updated and decisions are made based on information in both the Perception Array and the Prior Knowledge Library. As such, the Consciousness Event represents the awakening of the "conscious mind" of the decision element. Figure VII.1 shows the information flow to and from the Consciousness Event.

Information processing within the Consciousness Event logically follows the activities described in Chapter I for generation and updating of the mental model: input data, interpret data, perceive situation, generate and project courses of action, associate values with alternative courses of action, and select alternatives. Although these procedures were discussed in detail in Chapter I, it is appropriate to review them from a processing standpoint. The following example from TAC BRAWLER serves to illustrate information processing in the Consciousness Event.

Once activated, the Consciousness Event immediately makes a check to see if it has been self-called, that is, a pilot had previously planted it so that he could reconsider his current action at some future time. If the pilot has, for any reason, been conscious in the interim, the present

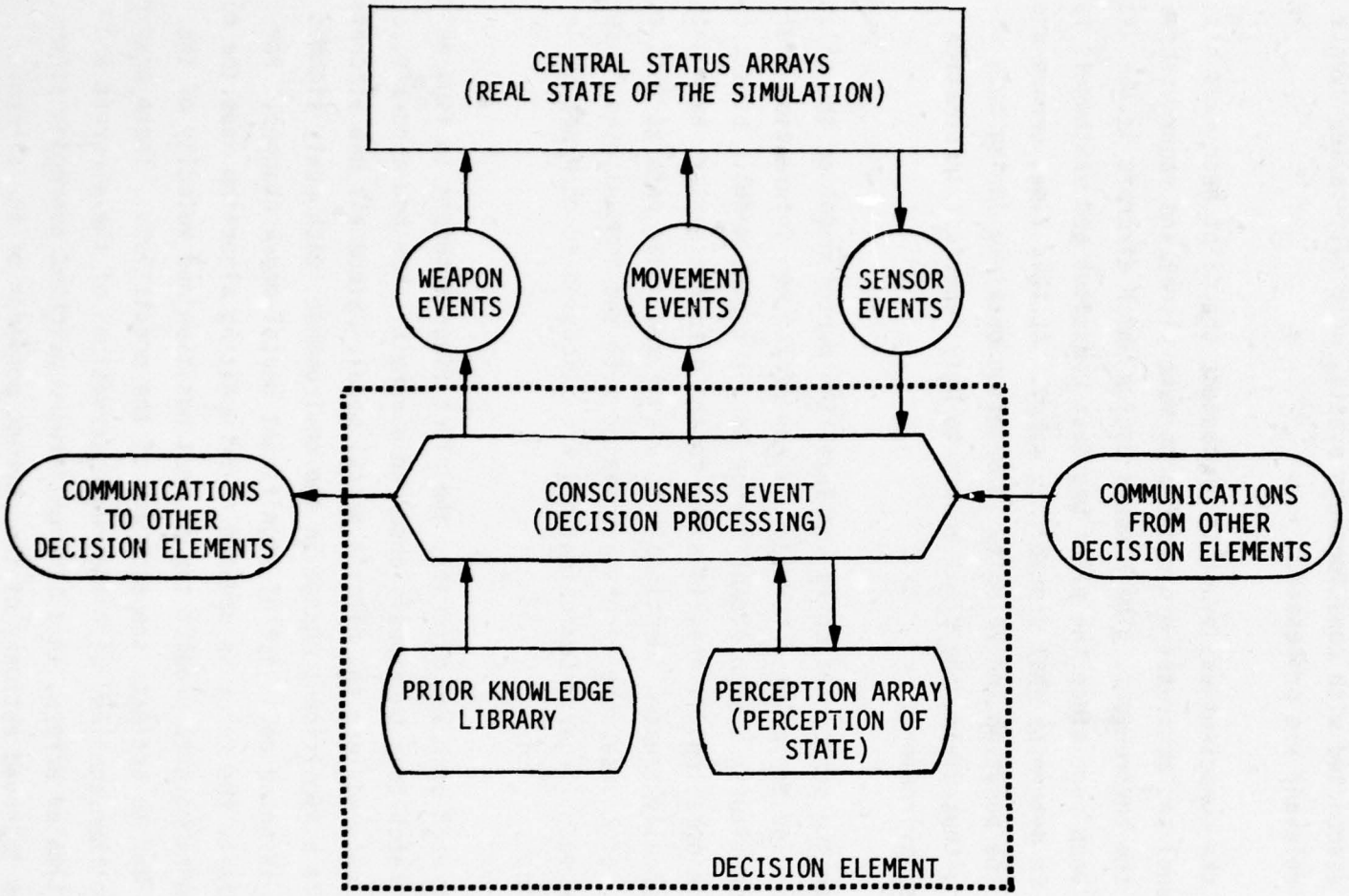


Figure VII.1. Consciousness Event Information Flow

event may be superfluous. In order to save the substantial processing overhead associated with consciousness events, only self-called events that are relevant are processed.

If the consciousness event is processed, the pilot retrieves his mental model and activates his sensors to make visual and radar observations of the environment. The sensors decide which aircraft in the vicinity have been seen since the pilot was last conscious and whether it is possible to determine what stores they carry. At this time, errors are added to the position and velocity information obtained during this observation, thus giving the pilot access to only imperfect information about his environment.

When the sensor search is complete, the mental model of the pilot is updated and the pilot determines if any of the new information available to him constitutes a significant change in his mental model. First, communicated information, if any, is added to the data the sensors have gathered. It is basically treated identically to visual and radar data at this stage, although, of course, the errors associated with the communicated "observation" are much larger. Communications are discussed in more detail in the next section.

For each observed aircraft, the pilot first attempts to find an identity match to a previously observed aircraft. For new aircraft, a track is entered into the pilot's mental model. Since all new aircraft constitute a significant change in the environment, each newly sighted aircraft is noted on a list of significant mental-model changes. For old aircraft, the track is updated. The updating algorithm uses the old mental model both to predict the present position and velocity of the aircraft and to estimate the accuracy of the predictions. These predictions are then combined with the new observations of the aircraft and their estimated errors, in a straightforward weighted averaging scheme, to get an improved estimate of the current position of the aircraft.

Essentially the same method is used with the velocity estimates, except that a third estimate, based on change of position, is also averaged in. These calculations also result in error estimates for the new mental model positions and velocities. If there is a large discrepancy between the predictions of the old mental model and the new mental model, a significant change (maneuver on the part of the observed aircraft) is considered to have taken place and the aircraft is placed on the significant change list.

After processing all observed aircraft, the pilot considers communicated value changes. These take the form of orders from superiors that modify the pilot's basic value parameters. Value modification is also discussed in more detail in the next section.

When he has finished updating his mental model, the pilot determines whether any unexpected changes in the environment require a reconsideration of the current action. If there are such changes, or if a long time has passed since the current action has been reconsidered, the pilot must decide on the best next action. A combination of preplanned logic and the data contained in the significant change list is used to generate a set of alternative actions. Each alternative uses the mental model to project its consequences, and the consequences are evaluated on an ordinal scale. The values obtained for each alternative are slightly randomized in order to simulate the imperfect judgment of the pilot. The amount of randomization is under user control. The highest scoring alternative after randomization is selected and the desired alternative is then implemented by constructing and planting appropriate events. Finally the Consciousness Event creates a new Consciousness Event that will occur when the alternative just implemented should be reconsidered, assuming no significant changes in the interim.

In reality, the monitoring of the real world and subsequent updating of a decision element's mental model is an analog process. Decisions are normally made when events occur that significantly change the decision-

AD-A060 402 DECISION-SCIENCE APPLICATIONS INC ARLINGTON VA F/G 12/2
VALUE-DRIVEN DECISION THEORY: APPLICATION TO COMBAT SIMULATIONS--ETC(U)
JUL 78 G L LUCAS, G F GORMAN, G E PUGH F49620-77-C-0089
UNCLASSIFIED DSA-67 AFOSR-TR-78-1398 NL

3 OF 3
AD
A060 402

END
DATE
FILMED
1-79
DDC

maker's perception of his environment. If there were no limitation on computational resources, one could adequately simulate this analog behavior by having all decisions reexamined very frequently regardless of whether there has been any perceived change in the situation. In practice, it is important to minimize unnecessary reexamination of decisions in order to keep the computational burden within reasonable limits. Using the consciousness event concept thus requires both a method for storing the mental models of each decision element that does not severely impact the data in fast access memory, and procedures for determining when a decision should be reexamined and for limiting the data retrieval and processing load associated with these decisions.

Since information required by the Consciousness Event for each decision element is needed only when decision events occur that involve that particular decision element, the mental models of all decision elements but the one currently being operated upon can be relegated to remote access memory. When a particular decision element "becomes conscious," the mental model for that particular decision element is simply called into fast access memory for processing. In TAC BRAWLER, all inactive mental models are stored in blocks in remote access memory and a series of block memory routines are used to fetch and return mental models to and from the memory. This technique for manipulating data is similar to the technique of "overlays" used for program sequences. Usually this technique results in a major reduction in the size of the central status arrays so that (even with very large simulations) it becomes practical to keep the central status arrays continuously in fast access memory. Even in virtual memory computers (such as the Honeywell/Multics System) this technique is effective since it eliminates unncessary pages of remote access memory.

As with many apparent solutions to problems, the data transfer technique described above presents another problem: namely, since Consciousness Events must occur at sufficiently frequent intervals to simulate analog decisions, how can we limit the amount of data transferred between fast access memory and remote access memory and thus reduce transfer times?

One method that has been used to minimize the data retrieval burden for Consciousness Events is to define several different "levels of consciousness," ranging from a minimal level that is adequate to handle routine, almost reflex, actions up through higher levels that are appropriate to more sophisticated decision-making. In order to bring a decision element conscious at any level, one can then retrieve only that part of the mental model that is required for that particular level of consciousness. An example from TAC BRAWLER may clarify this concept.

In TAC BRAWLER a simulated pilot may choose to fly to a specified route point, defined as being on a desired heading, at a desired altitude, straight and level, and at a desired velocity. Strictly adhering to the consciousness concept would require that the pilot "become conscious" at each significant maneuver point along the path (pull up, stabilize in climb, turn to desired heading, accelerate, level off, etc.). Clearly, if no significant event (such as the appearance of a target) occurs while carrying out this command, forcing the pilot to reexamine all alternatives fully at each maneuver point is a highly inefficient procedure. In TAC BRAWLER therefore, the pilot need only decide to fly to a specified route point and the actual trajectory is monitored and corrected at the pilot's lowest level of consciousness, which for TAC BRAWLER is the aircraft command level. This eliminates the need for conscious decision-making along the trajectory.

One should not confuse the concept of levels of consciousness with the multi-tiered decision element discussed in Chapter I. The multi-tiered decision element is a necessary technique for properly simulating a particular element which must make decisions at different decision levels. The TAC BRAWLER flight leader, for example, has the responsibility both for command decisions concerning the activities of his flight as a whole and for pilot decisions concerning the activities of his own aircraft. Although utilizing a similar data base (mental model), each tier uses separate logic and produces different action events consistent with the level of

decision necessary in the tier/command hierarchy. In essence, multiple tiers combine in one decision element the activities of two or more decision elements all using a similar mental model. Levels of consciousness, in comparison, are a result of efficiency considerations within a number of levels of consciousness, each with the same ultimate goal--maneuvering the aircraft for example. Rather than requiring complete processing of all information during the course of a Consciousness Event, levels of consciousness allow the use of selected portions of the mental model and selected processing algorithms to conserve computational resources.

In order to make the consciousness level concept operate smoothly, some mechanism must be provided to ensure that the decision element is brought conscious at the appropriate times in order to make the required decisions at the required level. In the example cited above, some method must be devised that would "warn" the pilot of a threatening target, if one should appear while enroute to his route point. In general, this requires that "consciousness alerting techniques" be developed.

There are a wide variety of procedures that can be used to accomplish the actual consciousness alerting. Specifically, each consciousness level can plant future events which will trigger a future consciousness event at the same level; lower levels of consciousness can plant events that will alert higher levels of consciousness; or sensor events can be used to trigger consciousness events. The real problem, however, is not the form of the alerting procedure (which can usually be accomplished by planting a consciousness event at some future time), but rather the development of efficient techniques for recognizing situations when a decision needs to be reevaluated. The need for effective consciousness alerting mechanisms has just recently been recognized, and there is therefore a need for further research into techniques that can be used to minimize unnecessary consciousness events without missing occasions when decisions are needed.

Some progress in the development of consciousness alerting techniques has been made, in spite of the inherent complexity of the problem. As was mentioned previously, the first action of the consciousness event is the processing of new sensor information. One method of reducing the computational burden of decision processing is to call sensor events at relatively frequent intervals, and use sensor information to trigger full consciousness events whenever a significant change has occurred. Significant changes would include new detections, loss of track of a previous detection, violent target maneuvers, and the like. Although the perception array for a particular decision element would still have to be transferred into fast access memory in order to test for significant changes in the perceived state of the simulation, most or all of the decision logic could be bypassed if no significant event is perceived to occur. In essence then, the sensor event can be considered as if it represented the lowest level of consciousness. It is the natural linkage for triggering the next level, and the perception array is certainly the repository of accumulated sensor data. Indeed, the sensor event almost must be considered the lowest level of consciousness since higher decision-making levels of consciousness require up-to-date sensor information prior to any decision processing.

In summary, decisions within a value-driven simulation are modeled using the Consciousness Event, which generates and updates the mental model of the decision element, and produces a course of action based on the updated mental model. To make efficient use of computer resources, levels of consciousness are used to limit the amount of data transfer and information processing within the Consciousness Event. To insure that unnecessary Consciousness Events are minimized, consciousness alerting techniques are used to trigger particular levels of consciousness, based on significant events in the simulation.

C. LINKING DECISION ELEMENTS

A complete value-driven combat simulation contains a large number (possibly hundreds) of decision elements, interconnected in a well-defined command structure, and each determining their individual courses of action

based on their own mental model of the real world (the Perception Arrays). In order to produce cooperative behavior between friendly decision elements and thus to produce a full spectrum of tactical options, methods must be developed to simulate information transfer between decision elements based on the chain of command.

For simplicity in discussing chains of command and their influence in the value-driven simulation, consider the command chain shown in Fig. VII.2, which describes the chain in the FIMOD air defense simulation. Three decision levels are shown: External Control, the Flight Leader, and the Pilot. Information is transferred between each level using a command language appropriate to each level. The use of explicitly defined levels in the chain of command and the use of a unique language to communicate between levels in the chain allows each level to respond effectively to the decisions made at the next higher level. Notice also that at each level of the command chain, decisions are based only on information legitimately available at that level. External Control uses information from GCI radars to provide flight vectoring information to flight leaders. The flight leaders receive the vectoring information and subsequently supplement it with information obtained from visual observations and airborne radar. When a flight leader acquires suitable targets, he uses a command language to make engagement assignments for pilots in his flight. The individual pilots assess the information from their flight leader and combine it with information from their own sensors to carry out their responsibility for engaging hostile targets. The chain is completed when the individual pilots command their aircraft and weapons to respond to their directives.

Explicitly modeling the command chain and associated command links effectively links decision elements into a coherent model of a fighting force. However considering only command chains neglects a valuable aspect of cooperative behavior; namely, communications between decision elements within a particular command level. For example, during the course of an air battle, different pilots will observe new hostiles at differing times. The first pilot to sight the hostile will normally convey this new sighting

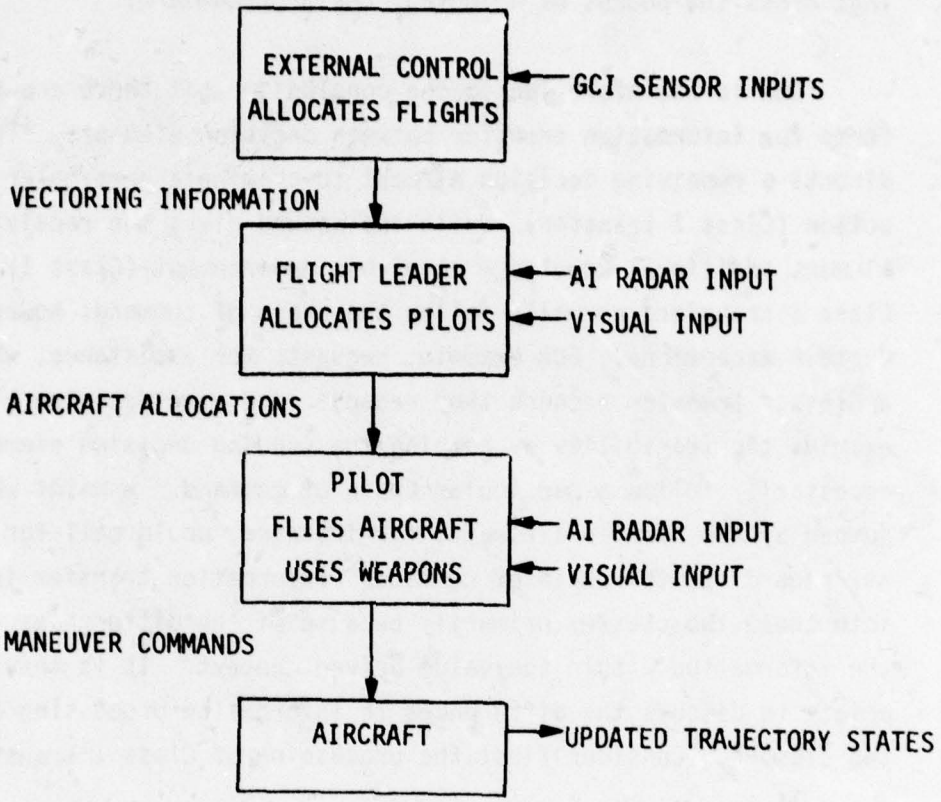


Figure VII.2. The FIMOD Chain of Command

information to other friendly pilots, so that they too may add the new target to their mental models. Similarly, calls for assistance and warnings cross the bounds of a logical chain of command.

One is therefore led to the conclusion that there are two basic forms for information transfer between decision elements. The first directs a receiving decision element to examine a particular course of action (Class I transfer), while the second gives the receiving decision element additional knowledge about his environment (Class II transfers). Class I transfers normally follow the chain of command; however, there are certain exceptions. For example, requests for assistance, while considered a Class I transfer because they request receiving decision elements to examine the feasibility of helping the sending decision element, do not necessarily follow a particular chain of command. A pilot who has been jumped by two hostile aircraft, for instance, would call for help without any regard for the chain of command. Information transfer is separated into these two classes primarily because of the differences in processing the information within the value-driven context. It is therefore appropriate to discuss the differences in information processing between the two classes. Consider first the processing of Class I transfers.

Perhaps the most obvious type of Class I transfer is an order that is passed from one level in the chain of command to the next lower level. In order to create a computerized representation of orders that retains the flexibility of the real-world situation, it is necessary to consider the true purpose of such orders. Orders are given to accomplish a coordinated action, rather than to literally "tell the subordinate what he is to do." Commanders do not, in general, have access to perfect information, nor can they perfectly predict the detailed development of the battle. Orders are therefore most useful when they take the form of guidelines to be interpreted by the subordinate. For instance, upon receiving an order to attack an enemy position, a company commander has to form a detailed plan of attack and he uses the order as a guide to select desirable actions, not as a replacement for his decision processes. If, upon attempting the attack, the

commander's company encounters heavy opposition, the commander could well halt his present action, at least temporarily. The important point is that the receiving element has considerations like the viability of the force under his command as well as the order. In the context of value-driven decision systems, the order determines a part, perhaps a very important part, of the value function of the receiver. If common-sense behavior is desired, however, it should necessarily replace all other pre-existing values.

This leads to the viewpoint that commands should be represented as a set of changes to the value function of the subordinate. Within this context it should be possible to obtain compliance with an order while still allowing the subordinate to act with "common sense," that is, to violate the order in certain situations. An absolute order can still be achieved by a sufficiently strong adjustment of the receiver's value function. The essence of the idea, though, is to make ths subordinate "want" to carry out the order by adjusting his values so that he will select decision alternatives which support the commander's intentions.

Many examples of orders of this sort can be found in the TAC BRAWLER simulation. For instance, if an element leader wants his wingman to adopt a certain formation he sends the wingman an order that specifies a "formation vector" telling how far behind, above, and to the side of the leader the wingman should be. This formation vector serves to parametrically control the formation-keeping portion of the wingman's value function. By itself, however, the formation vector does not tell the wingman how important it is to maintain formation in the face of an attack, or upon sighting a tempting target. Orders citing all possible contingencies are not feasible in real life; neither are they feasible in simulations. Generally, each portion of a decision-maker's value function has a scaling multiplier that controls the importance of its contribution to the overall value function. When this multiplier is set by a superior, it determines the "strength" of the order. In order for this scheme to work the value function must be

additive over the various aspects it considers. It is also useful that each portion of the value function, representing a different aspect, be bounded. In TAC BRAWLER an attempt is made to have each portion of the value function evaluate to a number between -1 and +1, regardless of how well or poorly that aspect is achieved by the alternative being considered. Only in this way can scaling of the relative importance of different aspects be readily achieved.

Orders control, then, both value function parameters and scaling multipliers. Furthermore, orders often need to contain additional information for the subordinate. Many orders have a finite time duration. For instance, if a flight leader allocates members of his flight to attack a flight of enemy aircraft, his orders would take the form of changes to the value of attacking a given member of the opposing flight. This is accomplished by making different enemy aircraft look attractive to each member of his own flight. Thus, he can cause his flight to properly spread its resources over the enemy force. Once the attack begins to develop it is likely that a pilot will have to place himself in a position where he will prefer, other things being equal, to continue the attack on his original target, but this is not necessarily going to be the case. In order for the situation to develop in a fluid manner, the order sent by the leader should not continue to constrain the subordinate as the flight develops. Orders need, then to be accompanied by time constants which specify the duration of order-driven value changes. If it is desired that the order last indefinitely, the use of an extremely large time constant will serve the purpose. In TAC BRAWLER, initial experiments with time constants use them to determine an exponential rate of decay of the order-induced value changes, but other forms, including linear decay and step functions, could well prove satisfactory.

One way of sending and receiving orders would be to design a set of command formats, the first word of which would indicate the format type. Special purpose computer code would interpret the remainder of the message and set the appropriate value-function parameters. This method may become quite cumbersome when many variants of command forms are desired.

A more general approach is obtained by placing all parameters which control value functions into a single indexed array. Elements in this array include parameters such as formation vectors, scaling multipliers, and even command time constants. Commands can then simply take the form of a string of index/quantity pairs, where the first element of the pair identifies the parameter to be altered, and the quantity portion of the pair gives the numeric value to which it is to be changed. This approach is used in TAC BRAWLER, with a generalization of the index concept to allow the indication of parameters that have different values for different objects in the mental models. An example would be the "value as a target" for different enemy aircraft.

From a computer processing standpoint, the transmission of orders becomes a relatively simple procedure. The sending decision element need only transmit sequences of pairs and triplets of data. A flight leader ordering his wingman to move to a new position in the formation would transmit three "value pairs" of the form:

(IX, XNEW)

(IT, YNEW)

(IZ, ZNEW)

where XNEW, YNEW, ZNEW specifies the new requested position and IX, IY, IZ reference the indices in the receiving pilot's mental model to which the new formation points correspond. To request his wingman to attack a particular hostile, the flight leader need only specify three "value triplets" of the form:

(IHOST, IVAL, VALUE)

(IHOST, ITAU, TAU)

(IHOST, ITIME, TIME)

where IHOST is the index of the hostile aircraft, VALUE, TAU, TIME are the new value for the hostile to assume, the time constant for the order, and the time the order was sent, respectively, and IVAL, ITAU, ITIME are

the indices in the receiving pilot's mental model that reference VAL, TAU, and TIME. The receiving pilot can thus consider these modifications to course of action during a Consciousness Event. Notice that by using this representation, no modification to the receiving element's processing logic is required, and if sufficiently motivated, the receiving element may disobey the order.

Class II transfers provide the receiving element with additional information about his environment, and are represented differently from Class I transfers. Included in this class are communicated sightings of new hostiles, information concerning friendly and hostile troop movements, weather updates, force readiness, and the like. Clearly there are a variety of different types of messages included in this class; however, the processing of information is greatly simplified if the communication originator is treated as if it were another type of sensor available to the receiving decision elements. The major difference between this new sensor type and other sensors in the simulation is that the new sensor has the ability both to preprocess the information it is receiving from the surrounding scene and to decide which other decision elements are to receive the information; i.e. it is a "smart sensor." By using this smart sensor concept, Class II transfers can be structured similarly to the structure of other sensor information transfers thus allowing a common logic to be used to process all incoming sensor information.

As an example, Table VII.1 describes the information transferred during a sensor observation in TAC BRAWLER, for the particular case of sighting other aircraft. Notice that aside from specifying the type of sensor,[1] all other information is independent of sensor type. This indifference to sensor type allows a common sensor processing algorithm to be used and simplifies the combining of information from different sensor types.

From the foregoing discussion, one can see that the primary consideration in the linking of decision elements is the realistic representation of actual combat command-and-control linkages. In order to provide for

[1] Sensor type is used only in the weight given to the quality of the information when information from more than one sensor is received.

TABLE VII.1

INFORMATION CONTENT OF SENSOR INFORMATION
IN TAC BRAWLER

- Index of detected aircraft (if known)
- Pointer to aircraft type (if known)
- Sensor type
    1. visual
    2. radar
    3. communication
- Time of sighting
- Position errors
- Velocity errors
- Afterburner flag
    0 - unknown
    1 - not in afterburner
    2 - in afterburner
- Store flags
    0,0 - cannot see stores, cannot identify stores
    1,0 - can see stores, cannot identify stores
    1,1 - can see stores, can identify stores

the necessary flexibility in the command-and-control structure in the simulation, it is necessary to carry out a careful functional analysis of the command-and-control functions that must be performed and to define the necessary input and output information associated with each of these functions. Once the basic functions are identified, it is possible to design generalized processing algorithms that can carry out the same function in a variety of command-and-control structures. To accomplish this, it is necessary to develop standardized information transfer formats that are to be used in the simulation. If the mental models of all decision elements are designed to interface with each other via these standard formats, it is then relatively simple to vary the C^3 links within the simulation and thus to simulate the flexibility of the real C^3 structure.

D. DATA HANDLING

The previous sections have discussed methods designed to efficiently model decision processing within a value-driven simulation. However, the real worth of the simulation can be tied not only to the efficiency of processing methods, but also to the efficiency of storing and handling the necessary data. As with any large-scale simulation, the value-driven combat simulation requires large amounts of data to operate effectively. Included in this data requirement is the need to store not only the real state of the combatants in the simulation, but also mental models of all decision elements, physical characteristics of the fighting forces, and detailed scenario descriptions. Many techniques have been developed to deal with efficiency considerations in large-scale simulations, and it would be beyond the scope of this effort to review each technique and discuss its applicability to the value-driven simulation. However, through the development of value-driven simulations a limited number of techniques have surfaced that directly relate to the usefulness of the value-driven simulation, particularly in the areas of data input, internal data handling, and output presentation. This section will discuss tested methods for efficient data input and internal data handling, whereas output methods will be discussed in the following section.

In order to provide as useful a simulation as possible, the problem of simply and completely specifying initial conditions must be considered as an essential part of overall system design. Users of large-scale combat simulations do not normally think in terms of the very detailed information that is needed to specify the initial state of the simulation. Rather, they are much more likely to think in terms of total numbers of key force elements and broad deployment concepts. The problem thus facing the system designer is to devise methods whereby the generalities specified by the user are transformed into the detailed specifications required by the simulation. Of course it is equally important to give the user the option to specify details when he so desires.

An example of one such input method can be found in the data language used in the FIMOD air defense simulation. In FIMOD, the user need only specify the basic scenario in broad conceptual terms (number and type of aircraft in the formation and the formation type), and the computer places each aircraft according to the formation type specifications (wedge, line, block, etc.). This procedure can be generalized to include ill-defined formations, such as tanks spread over a particular region, by specifying a distribution of forces over the region and having the computer determine the state details for each element of the force based on the desired distribution.

Similar to the FIMOD input data language is the TAC BRAWLER input procedure. In TAC BRAWLER, aircraft states are input as position and velocity vectors relative to the flight leader. The computer then not only determines the initial position and velocity vectors of all aircraft, but also additional state information required by the simulation such as initial angle of attack, throttle setting, climb rates, and the like.

Admittedly there are time penalties to be paid for using the computer to calculate the detailed input states of each force element. However,

use of this method not only reduces the input burden on the user, but also reduces input errors by keeping input data to a minimum. Furthermore, since the detailed input state specifications need to be computed only once (during initialization), run time penalties are minimized.

Coexistent with the efficient specification of scenarios is the desire to reduce redundancy in the specification of force element performance parameters. For example, normal procedure in a combat simulation involving, say, four similarly equipped aircraft may be to specify the performance parameters of all aircraft in one common block, the elements of which are of dimension four (indexing by each aircraft). Since the aircraft are all alike, and are similarly equipped, many of the elements of the common block will be identical. Such specification can easily strain memory resources when the number of force elements becomes large.

To relieve this memory problem, both FIMOD and TAC BRAWLER use a data pointer structure geared to reduce redundance in force specifications. Such pointer structures may be used in any large combat simulation involving large numbers of force elements. An example of a data pointer structure is shown in Fig. VII.3. In this example, there are four aircraft specified. Associated with each aircraft is a pointer that specifies (or points to) the memory location of a type data block. Each version of this type data block is shared by all aircraft which enter the simulation identically configured (e.g., aircraft 1 and aircraft 2). The type data block, in turn, points to data blocks which contain the actual detailed performance, radar, and weapons data, for the aircraft. These blocks in turn are shared by all aircraft having the same, say, performance data (e.g., aircraft 3 and aircraft 4). By using this pointer method, the user need only load data blocks for each type of data required, rather than for each force element in the simulation. For example, if a simulation contained 100 similarly equipped F-15's, the user need only load four data blocks--one performance block, one radar block, and, say, two weapons blocks--and the simulation would then contain one "type" of aircraft.

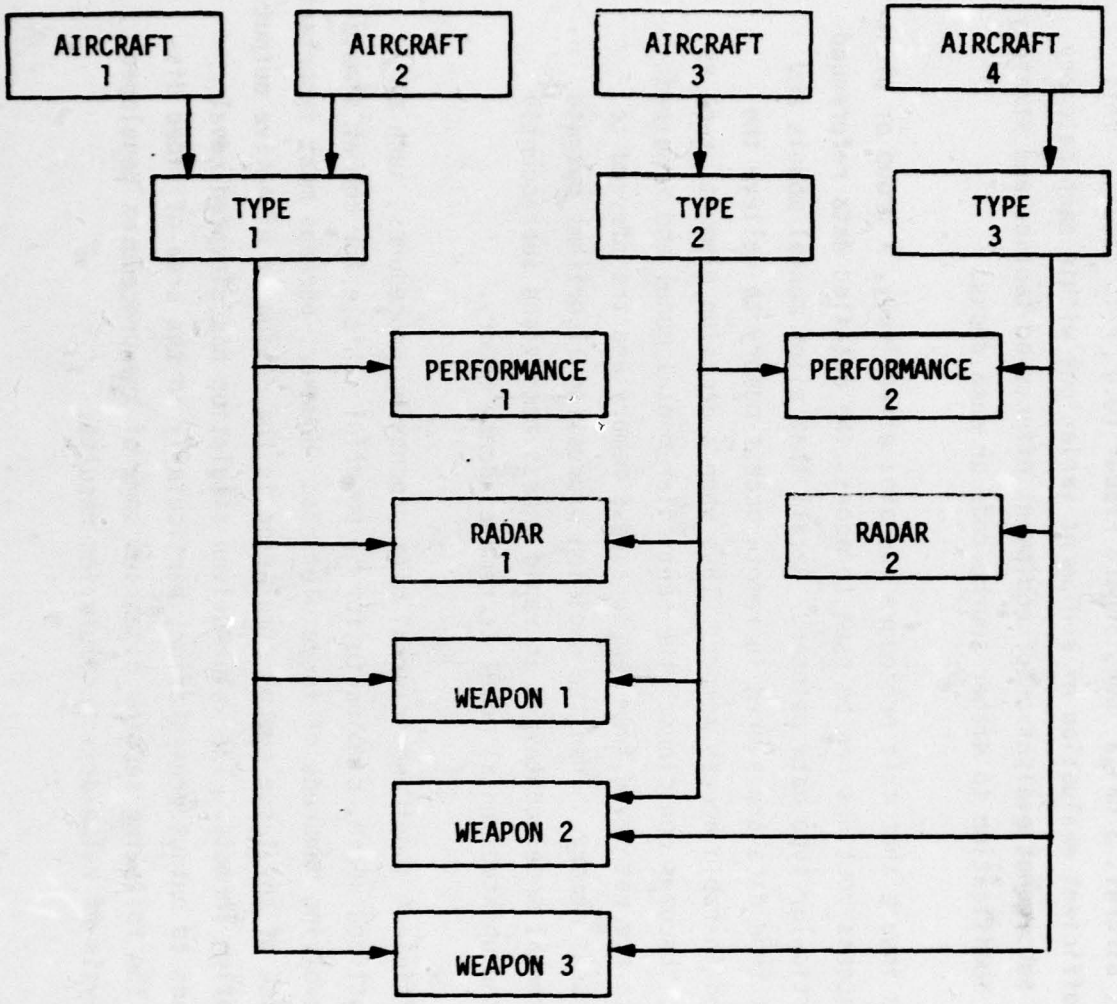


Figure VII.3. Example of Data Pointer Structure

The usefulness of this procedure extends beyond the reduction of data redundancy in that it allows comparisons of similar fighting units with nonstandard characteristics. For example, in Fig. VII.3, both aircraft 3 and aircraft 4 may be the same category of aircraft (e.g., F-16); however, aircraft 3 has a "nonstandard" radar (e.g., F-15 radar). This allows efficient evaluation of equipment variations within each category, and the subsequent evaluation of equipment effects on tactics and strategy, without modification to either source code or data decks.

To insure that this procedure operates effectively, a group of block memory access routines can be used to access the detailed data referenced by a particular type data pointer. Recall that pilot mental models and aircraft type data are stored in remote access memory to relieve the data load in rapid access memory. Thus when a decision element, such as a pilot, "becomes conscious," the particular mental model and required type data are retrieved from remote access memory and transferred to rapid access memory. Then the decision processing algorithms operate on the mental model contained in rapid access memory and subsequently return the updated mental model to remote access memory.

Efficient input and internal data processing procedures, such as those outlined above, combine to form a powerful vehicle for combat simulations involving hundreds of force elements. However, perhaps most important in the use of any large combat simulation is the design of effective output presentation formats. The value-driven simulation has dictated novel approaches to output presentation, particularly in the area of video displays. The following section discusses some of the procedures developed for analysis of value-driven simulation results.

E. THE VIDEO DISPLAY

A detailed combat simulation contains an immense amount of information that cannot be reflected in simple summary statistics. Very often, outcomes are not relatively simple functions of a small number of performance parameters, but rather are based on highly complex and interrelated phenomena--quality and quantity of information available to decision-makers, for example. Video display systems can be used to selectively interpret these complex processes and thus allow the analyst to gain a better understanding of simulation dynamics. Despite recent work in the field of computer pattern recognition, the human eye and brain still retain a decided advantage in speed and flexibility in the processing of visual images. Therefore, display systems can provide the critical link between the computational efficiency of the computer and the pattern recognition and analysis capabilities of the human brain.

This link is particularly important where the information-oriented simulation is concerned. In such a simulation, a decision entity bases his decisions on a perceived approximation to the real world (his mental model) and not on the real world itself. For the display software designer, this implies that displaying only real-world status does not present the true state of decision processing in the simulation, but rather only presents the ultimate outcomes of the decision process. To be totally effective, display software should reflect not only real-world status, but also the informational status available to a decision entity. The reason for this is fairly clear: a decision that seems "wrong" based on the perfect information assumption may be perfectly consistent based on the limited information available to the decision-maker. The task of the display software designer is thus to design a display system that can not only present the real state of the simulation, but also is capable of presenting the state of the simulation as perceived by a particular decision entity.

In light of the foregoing discussion, several criteria have been established for the design of effective display systems. These criteria specify that the display be human oriented, user oriented, and efficient. Although intuitively obvious, these criteria will insure that the display will present the pertinent information in an attractive fashion and that use of the display will be easily learned, operated, and interpreted. Each of these features is extremely important for acceptability and usefulness of the system; however, they are often neglected by the theoretical researcher.

To be human oriented, the display should exploit the pattern recognition capabilities of the human brain to the fullest extent possible. Stated in another way, the display should rely heavily on the effective presentation of visual (i.e., pictorial) information and minimize the presentation of alphanumeric data. The human mind has an incredible capability to absorb vast amounts of visual data and draw consistent and logical conclusions from that data. Even with the strides being made in computerized cluster algorithms and image processing, situation assessment is more efficiently handled by the human brain.

To be user oriented, the display should be tailored to the informational requirements and background of the user. The physical appearance and information content of a particular display plays a critical role in its usefulness as an analytical tool. An example will serve to illustrate this point. During the development of TAC BRAWLER, it became obvious that some means of displaying the physical air combat situation in a manner understandable to Air Force pilots would be necessary to order to evaluate the complex tactical decisions made by the simulated pilots within the model during "dogfights." Figure VII.4 shows an example of the display that was produced. In it, the scene from a particular airplane is shown including an artificial horizon for orientation reference, and "cross hairs" indicating flight direction. Targets in view of the selected pilot are displayed as "aircraft" (actually two intersecting triangles.) Hidden lines

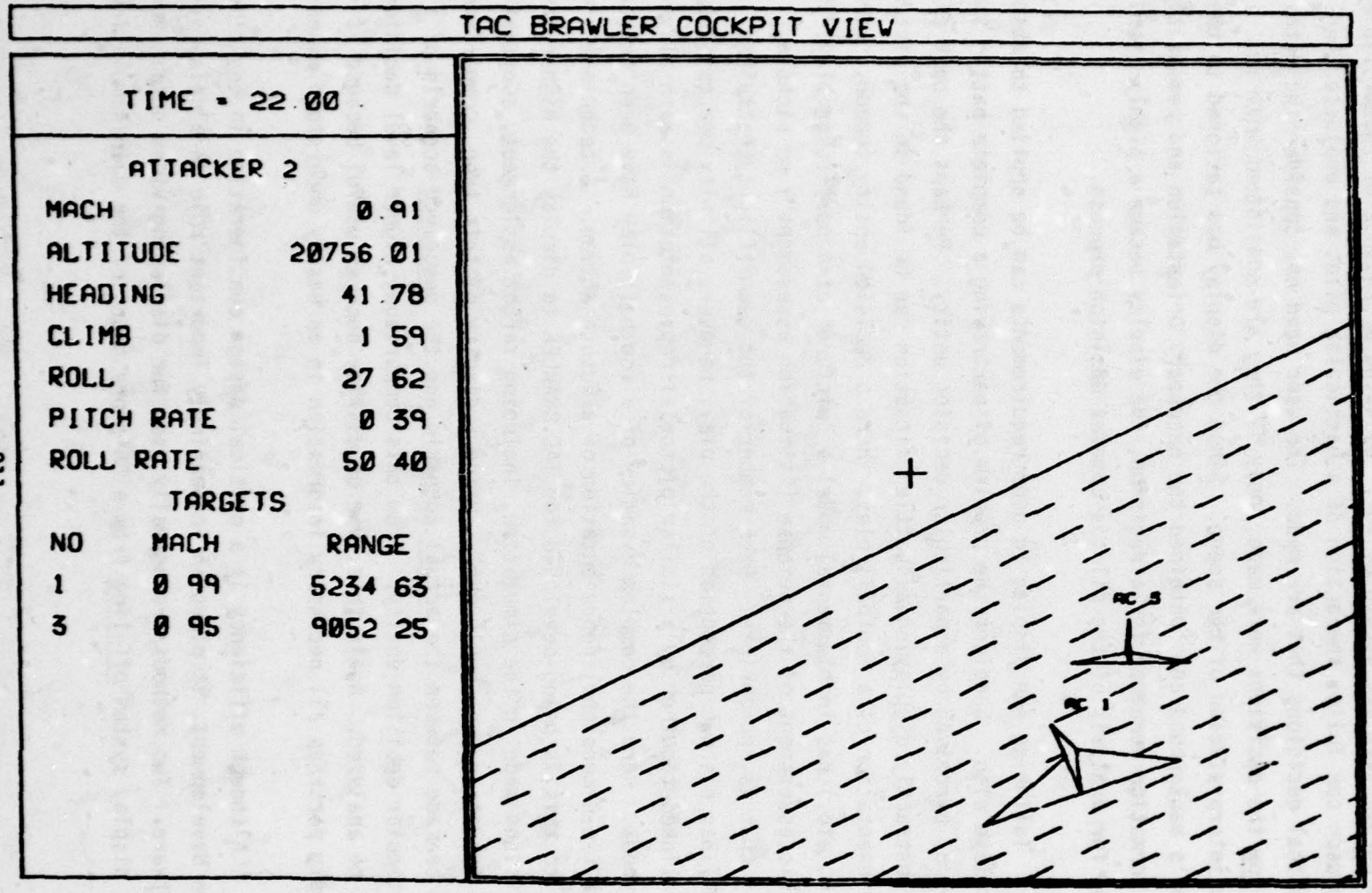


Figure VII.4. TAC BRAWLER Cockpit View

and perspective are accounted for. By using a sequence of these displays, the user can follow the action of a particular pilot and evaluate the tactical decisions that are made. The user need not consider the details of how the decisions were made, only if they are consistent with his own interpretation of the scene. Since the display was tailored to the user's background and contained the necessary orientation and perspective in formation required for a decision, the display became a highly useful tool for analysis of the pilot's complex decision process.

Tailoring the display to user requirements can be applied to abstract displays also. Consider the problem of portraying a complete battle situation as perceived by a particular decision entity. Perhaps the best example of abstractly displaying a "battle" situation can be found in the pictorial representation of a football play. Here a decision entity (coach, quarterback, etc.) has in his mental model a "picture" of a particular play. Based on his assessment of the defense ("situation assessment") he picks a play that is appropriate. Each member of the team (flight, battalion, etc.) has his own perception of the play; however, all play perceptions are linked together by a similar pictorial representation in each player's playbook. Here the complex dynamics of a football play have been reduced to a two-dimensional representation of a future action. A technique similar to this is being developed for TAC BRAWLER to display the higher level decisions made in the simulation, including target assignment, cooperation, and future weapons employment. Using a display of this type, comparisons can be made between the actual scenario and the perceived scenario of a particular decision entity. From this comparison, higher level decisions can be analyzed. Again this type of display becomes useful because it simply portrays all necessary information in an easily understood manner.

Although efficiency is a critical design consideration in any simulation development, it plays an increasingly important role in display system software. Two methods are normally used for displaying video data: running the display system off-line from a "history tape" of the main simulation,

and on-line in a completely interactive mode. Clearly, when run on-line, the algorithms used for the display should not be so complex as to dominate the running time or memory requirements of the combat simulation proper. To a lesser extent, off-line programs share this requirement. There is a tendency for software designers to be preoccupied with complex mathematical algorithms, information retrieval systems, and "clever tricks" in performing calculations. Although some of these procedures are necessary, the designer should not lose sight of the fact that the display is only a tool for data interpretation and complex procedures should be minimized.

Assuming the operating system contains the option for on-line versus off-line operation of the display system, the choice of which method to use can be based on two factors: urgency of the data and cybernetic resources required to present the data. For most situations off-line operation will suffice. However, when there is a critical need for on-line operation--such as when human intervention is required during the course of running the simulation--the efficiency of the display algorithms becomes of paramount importance.

Perhaps the only criteria that is allowed to break the efficiency rule is the desire to make the display dynamic. Dynamic software, as used here, consists of two concepts. First, the software should be capable of easy modification to account for changing user needs. This implies that standard modules (i.e., hard-wired) in the simulation be insulated from modules where requirements are most likely to change. For example, a change in input formats should affect only those routines that interpret input data, and should not require a change to any other system routine.

Second, dynamic software means that the user should have at his command several options for a particular display type. This criterion can be best illustrated by example. During the debugging stage of TAC BRAWLER, the cockpit view described earlier was used to determine

the adequacy of the pilot's tactical decisions. Since perspective was included in the display, at long ranges the orientations of the target aircraft were difficult to observe (due to the target's small size.) Thus, an option was built into the system whereby the targets were magnified while keeping all other relationships (angles, viewing orientation, etc.) constant. This allowed detailed observation of target orientation to be observed. Similar options could apply to viewing orientation, deletion of horizon, and the like, to present only that information that is pertinent to the analysis task at hand.

To summarize, display software should be human oriented, user oriented and efficient. Of these three criteria, perhaps the most important is that the display be user oriented. For, unless the potential user of the display can completely and quickly understand and interpret the displayed data, the usefulness of a display system diminishes quickly.

BIBLIOGRAPHY

Arbib, Michael A., The Metaphorical Brain, Wiley Interscience, 1972.

Collier, S., and G. Pugh, TAC COMMANDER: A Description of the Value-Driven System, General Research Corporation Report CR-110, May 1975.

Cooper, D., The Relevance of Problem Solving Methods in Artificial Intelligence to the Modeling of Command and Control, General Research Corporation, December 1977.

Everett, H., and G. Pugh, School Desegregation with Minimum Busing, Lambda Paper 68, December 1976.

Everett, H., "Generalized Lagrange Multiplier Method for Solving Problems of Optimal Allocation of Resources," Operations Research 11, 397-417 (1963).

General Research Corporation, Value-Driven Simulations and FIMOD, General Research Corporation Report IP-01-W, December 1975.

Gorman, G., and R. Kerchner, TAC FLIGHT: A Value-Driven Multi-Aircraft Simulation for Analysis of Close Air Combat, General Research Corporation Report 913-01-CR, November 1977.

Lucas, G., and S. Collier, TAC COMMANDER: A Description of the Intermediate Priority-Driven Model, Lambda Report 152, January 1975.

Lucas, G., and G. Pugh, A Modernized Plant Design and Scheduling System, Volumes I, II, III, IV, V, Merck & Company, December 1970.

NMCSSC QUICK-Reacting General War Gaming System (QUICK), National Military Command System Support Center, July 1967.

Noble, D., and G. Pugh, Staged Counterforce Exchanges, 1985-1990: Implications for Strategic Force Composition and Characteristics, GRC Report 904-01-CR, November 1976.

Pugh, G., "Lagrange Multipliers and the Optimal Allocation of Defense Resources," Operations Research, 12, 4(1964).

Pugh, G., and J. P. Mayberry, "Theory of Measures of Effectiveness for General Purpose Military Forces: Part I - A Zero Sum Payoff Appropriate for Evaluating Combat Strategies," Journal of Operations Research, Vol. 21, 867-885 (1973).

Pugh, G., "Theory of Measures of Effectiveness for General Purpose Military Forces: Part II - Lagrange Dynamic Programming in Time Sequential Combat Games," Journal of Operations Research, Vol. 21, No. 4, July-August 1973.

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER: AFOSR-TR-78-1398	2. GOVT ACCESSION NO. / 3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle): VALUE-DRIVEN DECISION THEORY: APPLICATION TO COMBAT SIMULATIONS.	5. TYPE OF REPORT & PERIOD COVERED: Final rept.
	6. PERFORMING ORG. REPORT NUMBER: DSA - 67
7. AUTHOR(s): G.L. Lucas, G.F. Gorman, G.E. Pugh	8. CONTRACT OR GRANT NUMBER(s): AFOSR F49620-77-C-0089
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Decision-Science Applications, Inc. 1500 Wilson Boulevard, Suite #810 Arlington, Virginia 22209	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: 61102F / 2304/A2
11. CONTROLLING OFFICE NAME AND ADDRESS: Air Force Office of Scientific Research/NM Bolling AFB, Washington, DC 20332	12. REPORT DATE: Jul 78 / NUMBER OF PAGES: 216
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office): 219 p.	15. SECURITY CLASS. (of this report): UNCLASSIFIED / 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

DSA-67

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

computer modeling, artificial intelligence, simulations, decision systems, decision-making, combat simulations.

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report, prepared for the Air Force Office of Scientific Research (AFOSR), describes the application of value-driven decision theory to the modeling of command, control, and communications, (C^3) in computerized combat simulations. Value-driven decision theory provides an unusually powerful method for representing C^3 in combat simulations, in that the theory provides an explicit representation of the human decision processes that drive the simulation. Each decision element represented in a value-drive simulation has its own internal model of the world which is formed by the integration of

Go next page

DD FORM 1 JAN 73 1473

393 157

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

20. Abstract continued.

of perceptions received via sensor and command links into a predefined structural model of the relevant portions of the external world. Employing its internal model, each decision element has the capability for generating and projecting alternative courses of action, for associating values or measures of worth with each alternative, and for selecting a preferred course of action. Through the use of a value-mediated command language, preferences can be conveyed among the decision elements of a command hierarchy or among different decision levels of a given decision element. This capability provides the decision elements the flexibility to override commands from higher level elements should circumstances strongly favor the adoption of an alternative course of action. This and other capabilities give the method the versatility for realistically representing the behavior of real-world C^3 systems in combat simulations. The method has thus far been most fully exploited in the development of a many-on-many air combat simulation in which the individual pilots are explicitly modeled and in the design of the C^3 system for the Air Force's Combined Arms Simulation Model (CASM). Noncombat applications are also under development.