Our cover reproduces an illustration from the May 30, 1857 issue of Harper's Weekly, in an article from their Borneo correspondent describing procedures used by the Dyak tribes in acquiring intelligence from their enemies. Although the illustration may seem a little silly in the context of a book on military applications of artificial intelligence, it does remind us of the importance and even reverence which has long characterized our attitudes about "intelligence." Reverence (and the closely related notion of fear) is a characteristic reaction to things that are strange and unpredictable. Is it perhaps useful to demand that this be an essential characteristic of any system which claims to exploit artificial intelligence?

If one merely demands that artificial intelligence is the operative word whenever an "expert system" consults a "memory bank," thereby implementing an appropriate "reaction" to a "stimulus," then it may be accused of retroactively plagiarizing cybernetics. At the same time it risks missing the new essence which is likely to be its primary raison d'être.

It is of course fundamentally illogical that a computerized system be unpredictable. Furthermore, predictability itself can be somewhat relative. Servo mechanisms used by naval guns in World War II to derive lead angles for targets are an example of an essentially predictable and understandable system. At the other extreme, the myriad of species that seem to have been generated by evolutionary processes is essentially unpredictable, even though they seem to have occurred as a series of reactions of expert systems to stimuli. Less cosmic examples are some of the beautiful computer plots of mathematical functions--mathematically predictable themselves but existentially unexpected and thought-provoking to the human observer. Particularly good examples are computer-generated "fractals"--families of functions and patterns so complex that they are impossible to appreciate without the aid of a computer. Complexity sufficient to generate unanticipated results is the fascination and promise of artificial intelligence.

Let us consider two aspects of particular relevance to military systems. One is the increasing impersonality that it can introduce to military operations. Bombers are being supplemented by air-launched cruise missiles and replaced by ground-launched cruise missiles (not to mention by ballistic missiles). Tanks already look like automatons, and are probably being tempted to "go all the way." The "Star Wars" nickname for the Strategic Defense Initiative (SDI) springs largely from the futuristic, computer-controlled battles that SDI contemplates. It would be cute if all this could lead to warfare in which the only casualties would be military equipment, but that is so far at best a philosophical speculation. A more popular concern is the night-
mare that these complex systems will somehow become Frankenstein's monsters with minds of their own—minds that might even initiate a nuclear exchange.

There may be a more likely and imminent hazard associated with artificial intelligence for the military. It is one that is subtle enough to be likely and dangerous. Refer again to SDI—whose deployed systems are sometimes estimated to cost trillions. The AI components of these systems could easily be tempted to become the major expense. This, by itself, would not necessarily be inappropriate. Certainly not if the resulting system would spell an end to the threat of strategic nuclear warfare. The hazard is that the AI components of SDI is already reasonably within the state-of-the-art. This is a great tribute to AI technology (both hardware and software), and it is not to say that implementation of the AI components of a space shield would not stretch the creative minds of the AI community nor require vigorous, continuing technological development. But there is a case that it would not require any unforeseeable technological breakthroughs. This means that AI work can be commissioned in parallel with and even ahead of the physical weapons that must be orchestrated. By the nature of things, the AI community may be confronted with a massive opportunity to develop these systems prematurely. It will be fascinating to see whether this community will find a responsible way to deal with these temptations.

In the meantime, AI has definitely "arrived" as a very significant and reasonably identifiable component of military operations research. It is an exciting component, and it definitely deserves to be vigorously pursued. The above concerns are real ones. Nevertheless, the raw materials of AI—new computer hardware, new software, and parallel processing—give us every reason to expect a decade of dramatic applications.

John D. Kettelle
Editor-in-Chief,
ORSA Softback Books
October 1986
PREFACE

The papers collected together as the Chapters of this monograph were originally presented at the Conference on AI in Engineering, sponsored jointly by ORSA and IEEE* and held at the George Washington University in Washington D.C., October 1985.

This monograph is the proceedings of the military applications papers of that Conference. The goal in publishing this set of papers separately was not strictly a thematic concern. While there is an abundant supply of scientific books on Artificial Intelligence (AI), there is simultaneously an extreme scarcity of good AI applications oriented books for military readers. This lack exists at the same time that the military-industrial-academic complex is showing much increased interest in AI. This was recognized by ORSA who hoped readers would find this monograph both timely and able to fill a much needed gap.

The chapters of the monograph offer a state-of-the-art view of military applications that is fairly well balanced in coverage of military topics, AI techniques, and mix of author backgrounds (military, industrial, and academic). Four papers are devoted to AI in military operations, four are devoted to AI in new military system acquisition, and four papers are devoted to AI and OR methodology issues. These papers are presented in Parts I, II, and III of this monograph, respectively.

This monograph offers a number of insights as to which AI techniques are actually being applied, how they're working out, and what experiences and lessons learned have occurred. While most of the papers describe either working prototypes or about to be implemented systems, none of the authors are "first time out" AI newcomers. Their papers do not cater to the computer programmer. Neither, do they offer a "you can build it at home too" flavor. The authors offer a useful state-of-the-practice overview.

This collection is essential reading for anyone about to begin a military AI application. It will also offer a varying degree of useful insights to more experienced practitioner as well. There are a number of questions that the careful reader can find answers to. For example, how well do rules work? Where and when do frames, models, and other knowledge representations find useful application? Which inferencing techniques are people actually finding appropriate and under what conditions? Is a LISP machine a prerequisite for good AI work? How can AI be integrated with operations research techniques and models? What's being done with user interfaces? And, of course, the most important question of all "Is anything useful likely to come out of all this AI activity?"

These and other questions are the ones that this book aspires to address. For many of these questions the debate is still open. If this book has any merit it is to try to help focus the debate a little better and to help continue the dialogue.

- B.G.S.
- W.F.H

# TABLE OF CONTENTS

**PREFACE**

**TABLE OF CONTENTS**

<table>
<thead>
<tr>
<th>PART I - Artificial Intelligence for Military Operations</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1) CAP/Tanker: A Model Based, Rule-Driven Expert System, ....... by Louis R. Gieszl</td>
<td>2</td>
</tr>
<tr>
<td>(2) Constraint Driven Distribution Scheduling, ....... by Robert Milne</td>
<td>12</td>
</tr>
<tr>
<td>(3) Operational Military Planning: An AI Approach, ....... by Gary Loberg and Gerald M. Powell</td>
<td>27</td>
</tr>
<tr>
<td>(4) Artificial Intelligence in Joint Deployment Planning, ....... by John W. Benoit, John R. Davidson, and E. Gordon Powell</td>
<td>56</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PART II - Artificial Intelligence in New System Acquisition</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>(5) ESB: An Example of AI Applied To Software Development, ....... by Bruce Blum and V.G. Sigillito</td>
<td>72</td>
</tr>
<tr>
<td>(7) A Lisp Machine Based Scenario Generator for the INNOVATOR Expert System, ....... by Arthur Murray, Costas Peggios, Meier Weinstein, and Barry G. Silverman</td>
<td>90</td>
</tr>
<tr>
<td>(6) JAMS: An Expert System for Electronic Warfare Vulnerability Assessment, ....... by Barry G. Silverman, Joel Simkol, Abid Kamran, and Charles Dublin</td>
<td>111</td>
</tr>
<tr>
<td>(8) Expert Systems in Mathematical Programming, ....... by Daniel E O’Leary</td>
<td>137</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PART III - Artificial Intelligence Methodology</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>(9) GENIE: An Inference Engine With Diverse Applications, ....... by Fred Brundick, John Dumer, Timothy Hanratty, and Paul Tanenbaum</td>
<td>149</td>
</tr>
<tr>
<td>(10) An Attribute-Referenced Production System, ....... by Henry L. Herz</td>
<td>161</td>
</tr>
<tr>
<td>(11) A Confidence Factor Extension to the YAPS Expert System Development Tool, ....... by Mohammed Nasiruddin, M. Srikanth, and George R. Cross</td>
<td>170</td>
</tr>
</tbody>
</table>
Part I - Artificial Intelligence for Military Operations

This part of the monograph includes four papers that focus on a single subject -- planning of military operations. While there is only one subject, two principal dimensions differ substantially from paper to paper: (1) the military echelon for which planning is addressed, and (2) the AI techniques utilized. It is these differences that are particularly instructive and revealing in this set of articles. There is as yet no optimal or standardized AI approach and each set of authors is clearly experimenting with and testing a different section of the "AI tool kit".

The military echelons addressed start at the air patrol level proceed to a battalion and corps level and wind up addressing inter-service, inter-theater considerations. At the same time the focus begins with single function planning (i.e., refueling logistics) and advances toward integration of all functions of concern to military commanders. These differences partially explain some of the reasons for the variations in AI techniques.

The differences in technique may be categorized by representation and inferencing topics. Knowledge representation schemes span the spectrum from rules to frames to plan elements to models. An equally varied set of inferencing techniques are described as well, including but not limited to backward and forward chaining, constraint propagation, and blackboard-based techniques. In some cases the use of techniques are impressive yet predictable. For example, the blackboard is utilized for the multi-agent planning problem in the third chapter while constraint propagation is applied to resource allocation in chapter two. In other cases, there are unexpected and innovative uses of more traditional techniques. As but one example, the fourth chapter exploits rule based technology quite beyond popularly held expectations of its limits. As another example, chapter one addresses the interfacing of rule-based reasoning with conventional simulation techniques.
CAP/TANKER
A MODEL-BASED, RULE-DRIVEN EXPERT SYSTEM

Louis R. Gieszl
Johns Hopkins University
Applied Physics Laboratory

ABSTRACT

CAP/TANKER is a program recently implemented at JHU/APL that has been proposed as a decision support system for the U.S. Navy. This implementation combined techniques of rule-based expert systems and event-store simulation. The program is an iterative event-store simulation written in BASIC for a personal computer. The structure of this program is that of a rule-based expert system; where the rules include control logic as well as domain specific heuristics.

1. BACKGROUND

One of the concerns of The Johns Hopkins University Applied Physics Laboratory is the design and development of future systems for the U.S. Navy. This includes hardware, software, and even operational tactics needed for the new system or subsystem. For example, a sizable ongoing effort here involves software and hardware options for the future. The hardware and its associated software tends to evolve in a natural way - taking advantage of new technology as it appears. However, the user interface problems are much more technology restricted. No matter how complicated a device is, you can always train someone to use it. But to make that usage easy or convenient may require more of a smart interface than can currently be implemented. Hence the increased interest in the last few years in "decision aids" or "decision support systems" to aid a decision maker - especially in interface problems with complex systems.

A few years ago, several prototype rule-based programs were developed at APL in an attempt to gauge the utility of such decision aiding tools for Aegis command staff. They were not developed beyond the prototype stage because the domain -- command decision aiding -- is clearly too knowledge intensive for current expert system technology to significantly augment. The lesson, of course, was to narrow the domain. But, the narrowing process was to be directed by need. The expert system development exercise provided this
lesson (initially given in any expert system development seminar, but probably always learned anew because of the unknown prior quantification of many domains) along with two other useful products. First, it developed a core of experience in rule-based programming -- with the tools on hand. Second, it provided examples to speak from; to show what could be done to handle less easily solved problems. This type of demonstration, along with the concerns of surface ship command broadening to cover Anti-Air Warfare (AAW) in much greater depth, led to the consideration of developing an expert system to help plan the needs of Combat Air Patrol (CAP) aircraft -- in particular, the logistics of in-air refueling.

The logistics involved in supporting CAP aircraft, especially when stationed long distances from the carrier, have traditionally been determined using "back of the envelope" calculations involving the number, type, and range of the CAP aircraft needed along with the number and capacity of the tanking aircraft available. A knowledgeable planner could make a few calculations and determine the rate of fuel usage, and hence the rate of tanker usage. An operationally experienced CAP planner could transform these rates to acceptable tanker schedules. The need, however, for CAP tanking expertise is shifting further away from the carrier as the responsibility for CAP deployment becomes one more facet of overall composite warfare command. Thus, the capture of this expertise in a portable computer program may provide a means for better distributing operational CAP planning knowledge. This was the motivation of CAP/TANKER, the system described in Reference [1], and the subject matter of this paper.

2. DOMAIN -- COMBAT AIR PATROL LOGISTICS

The domain of CAP/TANKER is the use of Combat Air Patrol (CAP) aircraft by the U.S. Navy and the logistical problems associated with the necessary in-air refueling over extended ranges and long durations. The amount of fuel burned by a particular aircraft is mainly a function of its speed; the endurance of an aircraft is a function of available fuel. The objective of the CAP/TANKER program is to determine the minimum number of tankers that will be necessary to sustain an input CAP configuration in some expected steady state of fuel consumption. This involves determining the appropriate start times of various tankers so that their recycling will create a "pipeline" providing enough fuel to support the configuration for an input duration time. For example, it
might take a "pipeline" of three tankers, spaced an hour
apart in starting times, to support a two-ring configuration
involving four CAP aircraft (at input ranges and coverage
angles) for a six hour mission.

In particular, the problem may involve a large number of
aircraft configured in rings centered on the carrier. The
number of aircraft, the spacing of the rings, the separation
of the aircraft, the coverage angle provided at each ring,
the average speed of the aircraft, the capacity and various
warning levels of fuel for the aircraft, the tanking proto-
col, the maximum number of tankers that may be used and
their capacity, consumption rate, and refuel timing, are all
inputs to the solution of this problem. A basic motivation
for this program as a decision aid is that very precise
planning of CAP positions and tanking schedules allows the
necessary CAP dispositions and control that are essential
for successful AAW defense -- especially when considering
the expected conditions of jamming, long distances, and
general autonomous operations.

For example, consider a CAP plan involving six aircraft --
two at a "medium" range (50 miles from the carrier) and four
at a "long" range (100 miles from the carrier). The various
(hypothetical) parameters input are:

Coverage Angle = 60 degrees
CAP Fuel Capacity = 25,000 pounds
Tanker Capacity = 40,000 pounds
Burn Rates = 50 lbs/minute for CAP & 40 for Tankers
Speeds = 20 miles/minute for CAP & 10 for Tankers.

The CAP are expected to proceed to their stations and orbit
at least six hours. The medium range aircraft will be
refueled as appropriate in order to be ready to relieve the
long range aircraft when they have reached a predetermined
reserve fuel level. The relieved long range aircraft will
then effectively become medium range aircraft -- ready for
refueling and relieving a succeeding long range aircraft.
CAP/TANKER use will be demonstrated with this example below.

3. EXPERT SYSTEMS AND EVENT-STORE SIMULATION

An expert system, one implementation of artificial intelli-
gence (AI), is a computer program that emulates human behav-
ior -- its performance, if done by a human would be termed
intelligence. Gevarter (see page 59 of [2]) defines it as:
"A computer program that uses knowledge and reasoning techniques to solve problems normally requiring the abilities of human experts."

It is expected to reason using dynamic chains of logic, and to explain its reasoning. An expert system makes use of a knowledge base, which contains heuristics (rules of thumb) as well as facts. An "inference engine" allows the logical traversal of this knowledge base from user inputs to its final advice. It may start from input states and work forward to an appropriate conclusion or result. It may start with a state and work backward to deduce causes. Or, it may be a combination. Most expert systems are rule-based; that is, the knowledge is expressed in distinct rules. Of course, some of the parameters may be in a traditional database.

A major theme in the original work on AI systems was the notion of "generate and test" (see [3]). The program is supposed to generate some hypothesis and then test the validity of this hypothesis in some "intelligent" manner. For example, does it solve the problem and satisfy the given constraints? Rules are used to represent the knowledge in a modular fashion -- that is, new knowledge can be added without rewriting the program. If the test can be expressed in a set of rules, then these rules can be considered as more knowledge -- the desired context to accept conclusions (maybe even filtering constraints on the antecedent of the rules placed in the knowledge base -- when the system is transformed into an efficient program), or simply additional rules for control (sometimes called metarules). The similarity of this notion to modular simulation (at least for forward tracking logic) seems clear. Discrete simulations can be written to emulate an expert or a process in the same sense of generate and test (although most simulations just generate). Modular simulations can be written to be driven by sets of rules (for example, decision tables). The event-store technique is an old method for maintaining a (predictive) blackboard with forward chaining logic.

The topic of this paper is not simply the description of another expert system, but is, in fact, the description of a merging of techniques in light of the current state of computer science. This merging involves rule-based processing and the type of simulation that has been a part of warfare analysis at the Applied Physics Laboratory since 1957 (see [4]). To avoid parochialism in the notions of simulation I will not reference any APL papers on this
subject, but instead will refer to a widely accepted source -- Alan Pritsker.

One definition of simulation that may lend credence to my position that expert systems were not simply created from LISP hacking or academic theories is found on page 6 of [5]:

"... (simulation is) the representation of the dynamic behavior of the system by moving it from state to state in accordance with well-defined operating rules."

In particular, event-store simulation has knowledge about the system's operation embedded in rules that store the events that "move" the program from state to state. This is stated (page 66 of [5]) as:

"... a system is modeled by defining the changes that occur at event times. The task of the modeler is to determine the events that can change the state of the system and then to develop the logic associated with each event type. A simulation of the system is produced by executing the logic associated with each event in a time-ordered sequence."

A review of the problems in trying to determine a schedule of tanking aircraft simply by manipulating the algebraic relations of ranges, consumptions, and tanker capacities brought out the fact that the dynamics of the system were not adequately represented. The use of more complex linear programming was even suggested to try to better represent the problem. That was quickly seen to be too cumbersome and inflexible. The need for an event-store simulation for this problem seems obvious. It was not obvious, however, that the rules that store the events could be expressed in the sense of expert systems. The object of this paper is to show that this can be done; and, in fact the event storing/retrieving mechanism is an inference engine -- using (predictive) forward chaining.

4. THE PROGRAM

4.1 The Program -- Architecture ... Computer Scientist's View

The general architecture of CAP/TANKER is shown in Figure 1. It consists of an executive (that handles initialization
too), a section for rules, a section for actions, and the inference engine (a control loop and the event-store mechanism). The rules and their corresponding actions are further refined to sections for each type of aircraft (and a "general" section for control rules/actions). The program is initialized with the desired CAP plan and aircraft characteristics. This sets up the proper arrays for an event-store simulation to model the CAP dynamics and fuel usage. The event-store mechanism acts as a time trigger to initiate a looping through a set of condition/action rules. The conditions are "if-statements" and an action is a set of instructions to store an appropriate event. When an event comes up as the next scheduled activity, the overall state of the simulation is updated and the rule interrogation is re-initiated.

![Program Architecture Diagram](image)

**Figure 1**

The program interacts with the user during initialization, during (input requested) periodic status reports, at each trial failure and restart, and on final output. This notion of failure and restart is my approach to an explicit "generate and test" paradigm -- somewhat unique in such explicitness in a discrete simulation. This is a (crude) form of backtracking - going all the way to time zero - but, it is effective for this problem. For a more complex problem, partial backtracking is certainly possible using event-store techniques. It would have some overhead in properly marking and "un-doing" events (see [6]).
4.2 The Program -- Dynamics ... The User's View

The dynamics of the use and evolution (updating) of CAP/TANKER is shown in Figure 2. The user interacts with the program by specifying his problem and receiving the computed advice. The expert interacts with the program in two ways. First, he may execute it as a user would to find areas of disagreement. Second, he may input new rules or change old ones to make the program's advice consistent with what he would give under the same input conditions.

**CAP/TANKER DYNAMICS**

![Diagram of CAP/TANKER dynamics](image)

**Figure 2**

The program initiates a discrete time (event-store) simulation of the CAP configuration input, tries to sustain this plan with one tanker, and then appeals to its rules (in the knowledge base) to determine what to do upon failure (an aircraft was not refueled in time). This inductive process is repeated until the total available tankers is exceeded, or the tanking plan being tried is found to be adequate. Some initial logic could be inserted in the program to allow it to start with an initial tanking plan more associated with a predicted fuel usage (vice starting at one tanker and incrementing), but this logic may be questioned by an individual user (unless it was his logic). The incremental approach seems less prone to argument, and is not very time consuming in practice.
Since the objective of this program is to find the necessary parameters (number of tankers and their scheduling) to create an automaton (a predetermined set of events) that can emulate a steady state CAP configuration, an immediate result of a successful trial is the ability to print out a detailed schedule by simply executing this automaton. This is also the mechanism to provide an explanation. The program may be run in a minimal output mode looking for a tanking plan. When one is found the program may be re-run for that particular trial with detailed output -- giving all the rules used.

4.3 The Program -- Example Problem

Typical usage of CAP/TANKER is best shown by example. We'll use the hypothetical example referred to earlier; that is, trying to find the least number of tankers to support two medium range and four long range CAP aircraft (with the previously specified parameters). An actual step-by-step review of the interaction of the program and the user should point out the benefits of model-based rule-driven programs.

Initially, for this example, the program tries to support the configuration with one tanker starting at T=5 minutes. This fails at T=310, so a restart rule ("start a new tanker 30 minutes prior to last fail time") is invoked, and trial two is started. In this trial one tanker starts at T=5 and a second one starts at T=280. This trial also fails at T=310 (the second tanker does not have enough time to make up the immediate need of fuel). Now another restart rule is invoked ("if no time was gained from the previous trial, start with more tankers"). So, trial three starts two tankers at T=5 (admittedly not an efficient operational tactic -- but this is not yet a SMART expert system). This trial gets to T=325 before it fails. There is a gain in duration -- but not enough to satisfy the conditions of the other restart rules -- so, the previous rule is invoked. Trial four starts three tankers at T=5. This supports the CAP plan and is stopped as a success by the user at T=515.

Three tankers will support the sample problem, but the low level of expertise in CAP/TANKER (sending out three tankers to work with two inner ring aircraft) leads to the need to add more sophisticated rules in general -- or to re-run the program using the insight gained from this run. The latter is, of course, the appropriate route for this example.

The first trial shows that there are no problems until about T=300. Also adding another tanker at the last minute does
not help (an insight from trial 2). Therefore, a proposed tanking plan is input to the program (one of the input options) where the first tanker starts at $T=5$, the second one starts at $T=250$, and the third starts at $T=300$. This is clearly more efficient usage of tankers than starting with all three at the beginning. With this tanking schedule, the mission goal of supporting this CAP configuration for at least six hours is achieved. Since the fuel given is clearly related to the CAP aircraft fuel used (and that is a function of time), it should be noted that the fuel given in this last case is virtually the same as that given in the previous case (of course the tankers used more fuel in the earlier case).

5. SUMMARY

CAP/TANKER was initially written in FORTRAN for an IBM 3033 to be run interactively from a terminal. It was then translated to BASIC for an IBM Personal Computer. Most recently it was translated to the BASIC native to an HP 9000, to become one of the decision aids for possible use on a next generation ship. These languages were chosen since there are few ships that have access to AI work stations -- but they all have some type of computer that speaks FORTRAN or BASIC. There are less than 25 rules in this program of about 1000 instructions. It has fixed arrays (as expected in FORTRAN or BASIC).

The prime reason for developing this program as an expert system has been to provide a working prototype that could evolve into a practical tool in one of several directions. The current CAP/TANKER is such a working prototype. The planned evolution involves a more realistic simulation to use on the CAP configuration steady state problem (planning for enough tankers using average rates in a steady state automaton). Another possibility is to go from the steady state planning activity to a more dynamic real-time decision aid (this would, of course, include the steady state program as a special case). Further growth of this program and its usage is only limited by the amount of additional expert rule augmentation -- gaining expertise from further interaction with experts.

REFERENCES


Constraint Driven Distribution Scheduling

Robert W. Milne, PhD
Managing Director
Intelligent Applications Limited
Kirkton Business Centre
Livingston Village
West Lothian, Scotland
EH54 7AY

Formerly:
Chief Scientist
The Headquarters, Department of the Army
Artificial Intelligence Center
HQDA DAIM-AI, Wash DC 20310-0700

ABSTRACT

An expert system for the distribution of complex equipment to large organisations is described. This system is under development at the Army Artificial Intelligence Center and is an example of a frame-based constraint expert system. The system is designed not only to provide fielding plans for new items of equipment, but to allow senior decision makers to examine the effects of changes in the fielding plans. The system provides several advances over previous approaches. Most distribution planners are hard-wired to a single priority list. In this system, the priority list can be interactively re-ordered and the resulting fielding plan displayed. The system can use either explicit ordering instructions or high level advice like "training facility first" to establish its list. It is currently being enhanced to allow for the coordination of several fielding plans. For example, it can co-ordinate the distribution of trucks with major radio systems as needed. Another extension is the addition of data-dependency backtracking to allow for the changes in one schedule to be reflected in other schedules.
1. FRAME BASED CONSTRAINT SYSTEMS

In the beginning was the rule-based expert system. In these systems, knowledge is represented in the form of if-then or "production rules" [1]. The fact that all animals with hair are mammals is typically represented as follow:

IF the animal has hair
THEN the animal is a mammal.

In this knowledge was the power of the system. This representation was a significant step forward from traditional techniques because the knowledge was much easier to write down, easier to manipulate and easier for experts and novices to understand.

In a rule-based expert system, the control or inference engine was very simple. For the current hypothesis, it found all rules that could be used to prove that it was true. It then tried each of these rules in sequence. For each fact in the IF part of the rule, it again searched to see if it had a rule to conclude that fact. If not, then it would ask the expert who was using the system whether this fact was true or not.

Two types of rule-based expert systems were developed. Backward chaining systems started with a hypothesis and tried to prove that it was true. If the current hypothesis was not true, then it moved on to the next hypothesis. This works well when there is a smaller number of hypothesis than possible facts or questions. The other approach was forward chaining. If there was a large number of possible conclusions, but only a few facts, it is better to start with the facts. This approach can also be thought of as 'data-driven'.

This approach worked for simple problems. In the early days of expert systems, this was the only technique that was available to the field. Expert systems could be written, because the inference engine gave us a very simple paradigm for solving problems and greatly reduced complexity. In fact, what was produced was an algorithm as a way to explore all possible solutions to problems.
After a time though, problems became too complex for simple rule bases. Problems were tackled that used many types of knowledge, such as speech recognition. This led to the development of the blackboard architecture [5]. The blackboard is a way to control and allow for the interaction of many knowledge sources. The knowledge sources can be very diverse.

For example, in speech recognition, knowledge is needed about phonemes, syllables, words, phrases, sentences, and sentence meaning. Each type of knowledge was represented by its own unique knowledge source. The blackboard was used to provide a means of access and control between knowledge sources.

This also introduced the problem of a scheduler. With the presence of many knowledge sources, it was necessary to control when each knowledge source looked at the blackboard and what each was allowed to do. The scheduler was the first step towards a more complex inference engine. The field of expert systems then had enough experience in building simple systems that more complex control structures could now be used.

The blackboard architecture has proven to be very useful for military applications such as sensor fusion and command and control systems. Recently the blackboard concept has been expanded to include two blackboards, one for data and the other for goals [6]. This allows the system to reason at higher levels of abstraction, as is often necessary for planning.

Recently other types of problems are being looked at. Two new areas are Planning and Diagnosis. For planning, a type of expert system known as frame-based constraint systems (FBCS) is used. For more difficult diagnosis, Structure and Function expert systems [2] are becoming common. A frame based constraint system is dominated by two parts. Frames or hierarchical knowledge representation and constraints. Frames [7] are useful since they allow for the hierarchical decomposition of data and the mixing of procedural and declarative knowledge. The also allow for the inheritance of information, giving a very powerful means of using knowledge.
By using a hierarchical approach to the knowledge representation, it is easier to solve problems at many levels of abstraction. In many domains, the constraints can be introduced one level at a time by stepping down through the frame hierarchy.

Many problems are dominated by constraints, especially in planning [8]. For example in planning the allocation of equipment, there is a constraint on the total number of items, the number of items each unit can transport, the number of items each needs to accomplish their mission, the amount of supplies they have, the amount of training they have, etc.

The ideal way to plan would be to write down all the constraints and have the system generate all possible plans which satisfy them. The decision maker could then select the best plan for his purposes based upon some criteria. The problem isn't quite that easy though.

Frame based constraints can be implemented by having demons associated with the value of each slot. This demon checks anytime the value changes to see that it is consistent with some constraint. For example, one can make a particular value to always be the sum of two other values, or make one date two weeks after another date. If a value on a slot is constrained by enough knowns, then its value can be determined.

This is the technique of 'propagation of constraints' [9]. For example, one knows that the output of a digital AND-GATE is constrained to be ONE when both of its inputs are ONE. If both inputs are known to be ONE, then the value of the output can then be assigned [10]. If a a set of values can be assigned to all slots which meet all the constraints, then we have a successful plan. In a tightly constrained problem, we can fill all values by just propagating the constraints. Some problems are under constrained, that is there are many possible solutions which satisfy the constraints. In these problems, the constraints are used to greatly reduce the search. Some problems are over constrained, that is, it is very hard to find a solution which meets all the constraints.
In these problems, search may be needed in order to try many alternatives. Data-driven backtracking can be used to greatly reduce the search as well.

One can think of a search algorithm as a generator of possible daughters, followed by a trimming stage. Later in this paper this will be discussed more fully. Constraints are most often used to trim next possible daughters, by eliminating those that don’t meet all the constraints. For example, if the total path of the solution must be less than 300 miles, then all generated paths longer than this can be trimmed. Some times there are global constraints, such as "the route must stop at a gas station". These can be enforced by making sure the value of the slot in the frame has an agreeable value.

Constraint driven planning problems are very common in AI based decision support systems. At the ARMY AI Center, such a system for the distribution of equipment is under construction.

2. INFORMATION MANAGEMENT

At the Army AI Center, the concerned is with the application of AI technology to the management of information. The focus is on ways of making information more accessible and ways of turning data into information. The primary interest though, is in a decision support system for equipment fielding. Of secondary interest are systems, such as natural language interfaces, that provide the ability for "computer shy" managers to examine their data.

There are several areas where computers and AI technology can assist with information management. These are computer graphics, natural language techniques and expert systems.

Of primary importance to the information manager is graphics. Graphics sell!! Since the role of these systems is to present information, graphics displays are essential. Many useful systems are 90% graphics and 10% computation. Most of the facilities are to allow someone to display their results and examine information. This is part of the local philosophy for information management. Display the
information so that the decision maker can see the implications of the possible choices, but don’t make the decision for him. It is easy to underestimate the value of graphics. Many application areas have been found where the user did not need an expert system, but rather needed a robust way to display his information.

Of primary interest are systems that let decision makers ask "what if" questions. At the simplest level, a simulator goes a long way for this, but automatic "what-icers" are better. Another feature is the importance of real data for the systems. To be at all practical, they must be able to access large databases (>1 million records each) and use this data efficiently.

2.1 The Golden Shield

One effect that this, and many similar, projects have is to produce many useful side effects that have nothing to do with AI. If a working system is never actually implemented, the effort will still have been worth it. This is due to a concept called "The AI golden shield". The promise of AI technology allows us to attack problems which looked too hard with traditional technologies. The final solution may not need AI techniques, but it would not have been discovered if an AI approach wasn’t used in the first place.

Using the Golden Shield, many hard problems have been addressed. At least they seemed hard at first. In fact most seemed so difficult that no one had tried them before. But with the Golden Shield, the problems were confidently attacked knowing that there were considerable computer science capabilities to draw upon. This is seen as the start of a trend. With the increased promise of AI, many more organisations will be trying automated solutions. Although only some of these will require AI, many more problems will at least be tackled.

3. AUTOMATED DISTRIBUTION AND RE-DISTRIBUTION SYSTEM

The basic system is the Automated Distribution and Re-distribution System (ADRS) [3] [4]. ADRS generates and displays distribution plans that reflect Army policy,
readiness considerations, and logistical constraints. The initial application domain for ADRS is signal (communications) equipment planning. ADRS has been designed so that planning for other equipment can be easily accommodated. Actually, ADRS is a general planning system that can be extended to other Army domains, such as personnel, in a straightforward manner. ADRS allows distribution plans to be coordinated so that users can see how a change in one plan affects other dependent plans. ADRS can also create redistribution plans for the older equipment that is to be displaced by the arrival of new equipment.

ADRS takes as input the schedule of when the items are available from the factory, and a priority list of units. The initial task is simple. When enough items have arrived from the factory, give them to the highest priority unit. The output of the system is a fielding plan of which units will get the equipment and when. This system is working with the new Mobile Subscriber Equipment (MSE) for the Signal Corps. This is a cellular radio system for the future battlefield. It should be noted that there is nothing in the current system that is dependent on MSE equipment. It is believed that it could be used to field any type of equipment.

The only problem is that this system may not replicate the way the world really works. For example, the priority list is changed very often. Due to a crisis, economic necessity or funding constraint, it may be decided to send the next piece of equipment to Japan. Or it may be decided to send the equipment to Germany, then Korea, then Asia. The next month one may want to send them to Korea, then Asia then Germany.

In the ADRS system, the priority list can be changed with a few clicks of the mouse. The commander can select which units he wants first, second, etc. and then the system can produce a new fielding plan in about 10 seconds. Currently, this takes about a week by hand.

Once the fielding plan has been produced, the system produces a color chart of the plan, showing when each unit will get each piece of equipment. Time slices can also be
made by month and year to show the level of readiness at each point in time.

The commander then uses the mouse to select a new priority list and can also display this. Finally, he can display a differential graph of the two plans and examine as many different plans as he wants. It is then his option to decide which plan he wants. In this way the user can explore and examine the effects of his ideas. The system presents the information to the user, and the user makes the decisions.

One example of the benefit of this: using the normal Army priority list, the Signal Training school is the last place to receive the MSE equipment. This is because it is not an operational center. However, they are the ones that provide training for the equipment. With the ADRS system, it is possible to re-do the list and move them to first. Under the old approach, this would not be discovered for perhaps a year. It is clearly preferable to spot the problem instantly on a color screen.

4. SYSTEM ARCHITECTURE

The ADRS planner/scheduler generates distribution plans that reflect Army policy, readiness considerations and logistical constraints. Different Army policies, readiness considerations and logistical constraints have been codified into rules called "guidelines." By selecting existing or creating new guidelines, the user can determine all possible distribution plans that are allowed by a set of guidelines. By modifying the constraining guidelines, and comparing the resulting plans, a Army Planning Officer can interactively explore the factors that affect plans and thereby interactively construct a plan that best meets the needs of the Army. These distribution plans are stored in a database for later reference.

The basic algorithm of the system is "Generate and Trim". At each step, a list of possible candidates to receive the next item is generated. Each element of this list is then examined in turn. If all the guidelines present at that step agree with that unit, then it is given the assignment and removed from the list of candidates.
There are currently four different types of guidelines: "initial," "logical," "numeric," and "side effect." These guidelines all reside in the ADRS distribution guideline database. The initial guidelines are run once at the beginning of a scheduling session to select those units that are to receive equipment from the set of all known military units and locations. It is these guidelines that build up the possible candidate list. Logical and Numeric guidelines are used to guide the scheduler as it explores alternative schedules. The logical guidelines must be satisfied for the next step of a schedule to be considered at all.

An example of a logical guideline is the "signal school first". This guideline causes the signal training school to be the first unit to receive the new equipment. The guidelines have been developed in discussion with equipment planners and are the basic 'rules' of the system. Other example guidelines are "distribute to the European theater first" and "within a theater, distribute to its US based corp first", "once you start a major unit, complete it", and "break ties according to the priority list".

The Numeric guidelines return weights that measure how well a proposed next step of a schedule meets a guideline. A user-supplied schedule cost-function reduces the weights to a single number called a "step cost." The cost of a schedule is the sum of its step costs. This cost is used by the scheduler to explore the more promising schedules first. When numeric guidelines are in use, the system behaves exactly as the A* [1] algorithm.

Side effect guidelines are used to annotate schedules with database update operations that should occur in the context of a schedule. For example, if distributing new equipment to a unit implies that older equipment will be shipped out, then a side effect guideline would be used to update the assets of the affected unit by removing the older equipment. These side effects do not actually occur during the scheduling process, since multiple potential schedules are being considered in parallel. Rather, they are applied locally to modify the force structure database before the planner/scheduler works on a new distribution plan that depends on, or is to be coordinated with, the existing schedule. This allows the distribution plans of related
items such as tanks, tank fuelling trucks, and tank maintenance tools to be coordinated together. A change in any of these interdependent schedules can affect the total readiness of the units receiving the equipment.

The ADRS force structure database contains information about the military organisation’s units (components) such as name, superordinate unit, subordinate units, location, duty-status, current SRC, and current equipment assets. Guidelines access this information in the course of constraining the generation of distribution plans.

The graphics subsystem is responsible for producing any tables, timelines, or icons that may be needed to display individual distribution schedules, or the differences between various distribution schedules. There are both monochrome and color displays. Some of the color displays are controlled through a Control Window resident on the monochrome monitor. The graphics subsystem accesses distribution plans and information about the military components receiving equipment to produce a number of different displays. The simplest display depicts when units would receive equipment under a given plan. Other displays use Army-specific application-specific icons to present the state of the distribution at any particular point in time. The military organisations receiving equipment are displayed hierarchically, usually organized according to the Army’s organisation’s operational structure. This allows the flow of equipment into theaters, corps, battalions, etc. to be examined in detail. There is another display to indicate inventory changes that occur to units.

4. 1 An Example

Let us consider a simple example to show how the algorithm would work. Imagine the following guidelines:

Initial: Select MSE Units
Logical: Signal Unit First       then
       European Theater First      then
       Priority List Order.

The first step would be to execute the Initial Guideline. The "Select MSE Units" guideline will search through the
entire database of Army units and select those that are to receive the Mobile Subscriber Equipment. The predicate to make this selection is defined in LISP and is based upon the class of the unit, such as "Signal" and the unit size, such as "Division". The result of this guideline is a list of possible candidates, representing the units to be scheduled.

The system now proceeds to the Logical guidelines. It loops through each unit on the list of possible candidates and checks whether it meets the guidelines. On the first pass, only the Signal Training School meets the criterion. An assignment is made to that unit and the remaining units become the possible candidate list. In this simple example, Numeric guidelines are not being used, so there is no prediction of estimated completion cost.

On the next step, the system loops again through each unit in the possible candidate set. Those that meet the test of European Theater are placed on a list of possible next assignments. When this list is constructed, one has two lists: those units that meet the logical guideline and those that are waiting to be scheduled. The Priority List guideline then operates on the first list, by sorting the items. Any ties in priority are preserved in the form of a tie. At this point the system can now make an assignment to the first of these units.

The system then continues making assignments until its first list (those in Europe) is consumed. At that point the possible candidate list is examined to see which unit to assign next. In this case, the list is sorted by Priority order and the assignments continue as before.

5. FUTURE GENERATIONS

For the second generation of ADRS, the problem that equipment is not fielded in isolation is addressed. In order to field an item of major equipment, it is also necessary to make sure that you have the right unit present, with the right combination of MOS (military) skills. You must also have various items of support equipment such as trucks and generators and maintenance gear. When people are there, you also need real property,
barracks, etc.; services such as cafeterias; supplies for the equipment such as fuel and training ammunition. This cascades, in order to have the right unit, you must have the training system producing each enough people. Each of the above is in reality dependent on each other item, making for a very intertwined organization. The system can currently co-ordinate schedules in a pair-wise fashion.

In the second generation, there will be a cascade of systems. The MSE planner generates requirements which the other planners then meet. The system can be thought of as a simulator. When you distribute one item of equipment, it will simulate the possible effects, allowing the decision maker to see the effects of the current system on other systems.

There is another benefit of this effort beyond just the AI system. The Army has a three tier information architecture. The main databases are on Cyber or Amdahl computers, the information is then extracted to regional VAX class machines. Then the information is used on a local area network. This is being prototyped for Deputy Chief of Staff for Information Management. The information will be extracted from an Amdahl to be stored on the local VAX, and then used on a local area network of Symbolics Lisp machines.

The second side effect is that although the Army as a corporate body knows what supporting requirements are needed for each piece of equipment, very few individuals know. By bringing these requirements together in one place the information is institutionalized and it is possible to monitor its consistency. In other words, no one person knows what things must be taken care of in order to properly field an item of major equipment.

The third effect is that the corporate database (CDB) for the Army is being defined. The ADRS system will have to use the CDB to make its decisions. It’s reliance on the ‘real’ Army data leads to the development of this prototype and the exploration of the implementation of expert systems with massive amounts of data. The fourth effect is that this system is leading in the development of an Army Decision Support System (ADSS). This systems will be one of the first AI tools in the ADSS.
The third generation involves another connection with reality. The world never works out as above. In the second generation, it was assumed that each sub-schedule can meet the original schedule. What normally happens is that one discovers that one can’t get enough people trained, so we can’t produce the units needed to run these two pieces of equipment a month. The third system will follow these dependencies in order to follow the effects of changes. This is done by an AI technique known as data-dependency backtracking. For example, if we can’t train enough people to fill a unit, we cut the units in half. The unit was being distributed to use an item of MSE, so we can’t distribute it. That means that we don’t need some trucks, so that cuts out a shift at a factory.

When the system is built, you have a model of distributing the entire army – clearly too big for one small group to tackle. The project is being expanded bottom-up by building upon each prototype.

Another aspect is the distribution of old equipment. For new equipment, the availability comes from the factory. For old equipment, they form a cascade based upon when the previous equipment is freed. It is useful just to follow this cascade.

5.1 Constraint Languages

The above guidelines were presented as a set of constraints. Let us take another look at these guidelines. Each guideline represents a constraint on the next unit or units to receive the next item of equipment. Part of the development of the system is to make these constraints easy to specify. Initially the system had over 20 guidelines that a user could select from. Later examination however showed that there are really only three fundamental guidelines. These form a constraint language for the user to use. The user should be able to specify guidance in any way that is desired, but only using the constraint language.

For example, one guideline is "Do X first", where X may be the signal school, the European theater, or any other
sub-group. In the system, the user is able to specify this constraint and pick the X from a list of meaningful options that the system has. Another constraint is related to how to break ties. The example we have seen in this paper is the Priority List, but other options are possible. Again, the user selects the tie breaking option and the system presents him with a set of meaningful options.

By using this approach to build the user interface, it becomes very easy for the user to specify his guidance. In this system, the knowledge engineer builds the constraint language and the support behind each option. The user then describes what he wants in terms of this constraint language.

6. CONCLUSION

In this paper, the idea of a frame-based constraint expert system has been introduced. This type of expert system is ideal for planning problems and provides a more powerful framework than traditional rule-based approaches. The system described here, ADRS, is an example of this type of system. The initial distribution and re-distribution system (ADRS) was built under contract for the US Army AI Center by Scientific Systems Incorporated of Boston, Mass. and Smart Systems Technology of Mclean, Va.

7. REFERENCES


