

TACTICAL EVENT RESOLUTION USING SOFTWARE AGENTS, CRISP RULES, AND A GENETIC ALGORITHM

John M. D. Hill, Michael S. Miller, John Yen, and Udo W. Pooch
Department of Computer Science
Texas A&M University
College Station, Texas 77843
{hillj | mmiller | yen | pooch}@cs.tamu.edu

KEYWORDS

Military, Decision Support Systems, Software Agents, Genetic Algorithms, Crisp Rules

ABSTRACT

This paper discusses and identifies problems with tactical event resolution, one of the sub-steps in the *Course of Action Analysis* phase of the Army's Military Decision Making Process. Some related work is identified, a solution is proposed, and the design of a system implementing the solution is presented. Software agents representing biases of human planners develop reasonable allocations of combat effects. These allocations are examined through a genetic algorithm survival test based on a simple combat-results mechanism and a crisp-rules-based system to determine the "best" resolutions based on specific fitness functions. Information about the best resolution is then presented to the user. The paper concludes with a discussion of preliminary results and an evaluation of the system.

INTRODUCTION

The United States Army uses the Military Decision Making Process (MDMP) in the development of plans (Army 1997). One of the phases within the MDMP is *Course of Action Analysis*. Within this phase, tactical events involving portions of both the friendly and enemy forces must be resolved.

Military Decision Making Process

Battle staffs in the United States Army use the Military Decision Making Process (MDMP) to develop a plan that synchronizes the efforts of subordinate units to accomplish a given mission. In the *Course of Action Development* step the staff prepares several candidate courses of action (COAs) based on the mission and the commander's guidance. The *Course of Action (COA) Analysis* step (commonly referred to as "war-gaming") helps the staff determine the outcome of the COA. Validity of each proposed course of action is based on criteria including suitability, feasibility, and acceptability. Results from the war-

gaming process are used in the *Course of Action Comparison* step where the commander selects the "best" COA to be refined into an Operations Order (OPORD).

Course of Action Analysis (War-Gaming)

In this step the candidate courses of action are analyzed through a "war-gaming" process to determine if they are valid. In the war-game a representative from each of the battlefield functions (maneuver, fire support, etc) provides input about the expected results as the friendly courses of action are played out against the enemy courses of action. The results from each course of action will be used later in the *Course of Action Comparison* phase. The steps of the war-gaming process are:

1. Gather the tools.
2. List all friendly forces.
3. List assumptions.
4. List known critical events and decision points.
5. Determine evaluation criteria.
6. Select the war-game method.
7. Select a method to record and display results.
8. War-game the battle and assess the results.

Tactical Event Resolution

This project focuses on the eighth step, in which the critical events identified in the fourth step are resolved and the results determined. This is normally a manual, *ad hoc*, process where the forces and combat effects on each side are tallied and the Operations officer and the Intelligence officer determine the outcome.

THE PROBLEM

Tactical event resolution in the manual *Course of Action Analysis* phase suffers from several problems. These problems are primarily related to constrained time and to difficulty in communicating ideas between staff members. More specifically, the problems can be stated as follows:

(1) Time constraints of the manual process don't allow for more than a cursory examination of each tactical event.

(2) Communicating the tactical event resolution takes time, and it is difficult to create, share, and maintain a common understanding of how the event is resolved.

(3) Staff officers have different ideas about the methods that should be used to resolve tactical events. These individual biases and interpretations have to be explained and clarified, further slowing the process.

(4) The resolution of an event involves the consumption of available resources, such as direct fire strength, indirect fire capability, expenditure of minefield-breaching assets, and fuel and ammunition consumption. It is difficult for the staff to keep track of all of these resource expenditures.

(5) Typically, the staff just lumps all of the enemy and friendly capabilities together and decides the outcome, ignoring any ordering of engagements within the tactical event which might yield different results. The resolution of an event will involve the placement and movement of forces, both of which require a representation of space and time.

(6) Similarly, changes in allocation of forces or effects between engagements within the tactical event might yield different results, but are ignored in the aggregated approach.

(7) At its basis, tactical event resolution requires a fundamental combat results mechanism. Although many results mechanisms are available, staffs often just "wing it" without defining and enforcing a particular mechanism.

PROPOSED SOLUTION

Any system that is designed to make tactical event resolution better must provide capabilities that address each of the problems that have been identified. The entire process must be accomplished faster than the current manual methods, determine the best application of the available combat power effects, and incorporate in a transparent fashion the different methods and interpretations of the staff members. The solution lies in providing the following capabilities:

(1) Overcome time constraints with automated support for tactical event resolution. A great deal of the ad hoc manual process can benefit from automation.

(2) Provide a graphical user interface for quick input, common understanding, and visualization. This will help to ensure every staff member has the same understanding of what's going on.

(3) Incorporate selectable rules or the ability to select biases. This will allow different people's ideas about how the event should be resolved to be included in the system, and make them apparent to everybody else.

(4) As the system considers various ways to conduct the event it must be able to remember resource consumption and be able to return to a previous level if necessary. The system should use automation to keep track of everything, from resource consumption through intermediate results to conclusion of the tactical event. Ultimately, this system should provide the results to a higher-level planning system.

(5) The system needs at least a minimal representation of both space and time that allows the staff to consider the effects of time and space on the outcome of the event. For instance, the staff should be able to set up the event as one engagement, simultaneous engagements, sequential engagements, or a combination of sequential and simultaneous engagements.

(6) Provide a mechanism that examines the effect of different allocations of combat effects that is integrated with the ordering mechanism and can be guided by the rules or biases identified above.

(7) Segregate the fundamental results mechanism so that the staff can choose the one they think is appropriate. This will ensure that the same results mechanism is used consistently and uniformly throughout tactical event resolution and that every member of the staff can understand exactly how the results were determined.

RELATED WORK

There is a great deal of work being done in planning, decision-making, and command and control operations. Most of it is at much higher levels than the scope of this project. For example, the Army Modeling and Simulation Office (AMSO) has identified technology voids in the areas of automated decision aids, COA tools, and tactical information aids. (Delaney 1999. Personal communication.) This project could support, in a small way, all three of the areas mentioned.

The Army Research Laboratory (ARL) is generally focusing on developing the infrastructure to support command and control decision-making (visualization, software agents, collaboration tools, multi-modal interaction, etc.) (Emmerman 2000. Personal communication). ARL is also funding a research program in Intelligent Information Processing for Visualization (IDFL 2000). One of the IDFL projects, FOX-GA, is a tool that uses course-grained representations in order to provide timely COA generation and assessment (Hayes and Schlabach 1998). Its relation to this work lies in its use of a genetic algorithm for allocation of assets, but at the higher brigade COA level (Schlabach et al. 1998). FOX-GA will be transitioned to the Communications-Electronics Command (CECOM) to be part of the Command Post XXI Advanced Technology Demonstration (Slife 2000. Personal communication) (DARPA 2000).

Army Major Robert H. Kewley, Jr., combines fuzzy inference systems with genetic algorithms to form a fuzzy-genetic decision optimization (FGDO) system that he applied to the battalion-level tactical course of action (COA) development problem (Kewley 1999). In his system a fairly sophisticated tactical simulation module is used to evaluate the outcome of

proposed COAs. The performance of each COA is fed into a fuzzy preference module. From this module an overall fitness for the COA is fed back into a genetic algorithm module that continues to produce modified COAs. Kewley's approach differs from this project in that he focuses at the higher (battalion) level course of action and uses a sophisticated simulation. Naturally, it takes much longer to solve such a complex problem. This project, although it could be used at battalion level, is focused more at the individual tank or platoon level and uses a simple combat results mechanism. The similarity between the two projects lies in their use of genetic algorithms to determine better outcomes. Also, Kewley's project recommends future work on biasing the initial selections, which is a fundamental part of this project.

Genetic Algorithms (GAs) draw on the adaptive "survival of the fittest" capabilities inherent in Darwinian evolution. One fundamental aspect of a GA is an encoding that allows the description of every possible state of a system, but which is also amenable to rapid calculation. This encoding is typically referred to as the "chromosome," although the term "genome" may be appropriate if the encoding contains distinguishable sub-sections. Another fundamental piece is a "fitness function" which is used to decide how good the outcome of the system is when a particular chromosome is used. The algorithm creates an initial population of the chromosomes, possibly using heuristics to ensure a pretty good set. The fitness function is applied to each chromosome, allowing them to be ranked. As the algorithm produces each new generation, the more fit member of the previous generation have a higher probability of reproducing. Children for the new generation are produced by pairing two parents, and with some probability crossing their genes. Also, with a small probability, the children may experience a mutation in the elements of the chromosome. A seminal discussion of genetic algorithms appears in DeJong's dissertation (DeJong 1975). Goldberg provides a thorough presentation of GAs in his book (Goldberg 1989).

Software agents are notoriously difficult to define, since the title can be applied in many ways. Russell and Norvig define an *agent* as "anything that can be viewed as perceiving its environment through sensors and acting upon that environment through effectors" (Russell and Norvig 1995). Franklin and Graesser provide a taxonomy of agent types, of which software agents are one branch, and a description of agent properties. Among these properties are reactivity, autonomy, goal-orientation and temporal continuity (Franklin and Graesser 1997).

Rule-based systems allow knowledge to be represented as actions to be taken when certain conditions are matched. These heuristics, or "rules of thumb," are normally chosen by a domain expert and encoded by the developer. These rules allow abstract, symbolic approaches to be used in specifying knowledge based on human logic. CLIPS is a forward chaining LISP-like rule-based language that has inferencing and representation capabilities and is used to build rule-based expert systems (Giarratano and Riley 1989). CLIPS processes the rules by using RETE, an algorithm that solves the difficult many-to-many matching problem encountered when matching rules with facts (Forgy 1982).

SYSTEM ARCHITECTURE

The system architecture can be broken down at the highest level into three processes: event generation, analysis, and visualization of the event resolution. Event generation takes inputs from the user and converts them into an event description useable by the analyzer. Analysis uses a genetic-algorithm-based analyzer or a crisp-rules-based analyzer to resolve the event. Visualization of results is where the resolution of the event, and any intermediate information, is displayed to the user. A depiction of the major and minor components of the system (which also shows the details for the genetic-algorithm-based analyzer) is in Figure 1. The following sections describe the three processes in detail.

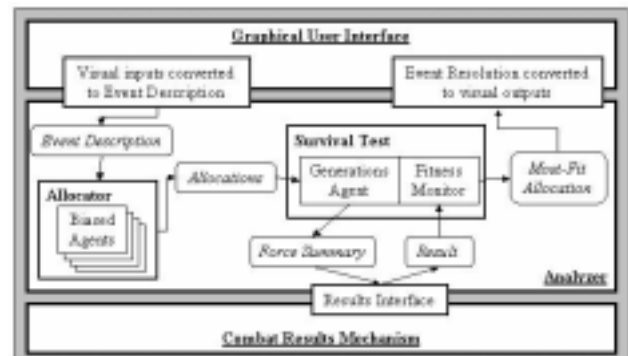


Figure 1: System Architecture (GA Analysis)

Event Generation

The event generation process begins with inputs from the user. The *Graphical User Interface* is a drag-and-drop display mechanism that allows easy interactive creation and setup of the events. The output of the process is an *Event Description*.

Interactive Event Creation. The GUI allows the user to define a static enemy (also known as the Red force) situation. In order to accommodate time/space ordering, the enemy can be broken down into any number of levels, where forces in the lowest level must be dealt with first. Also, within levels the enemy forces can be broken down into different forces. This is the mechanism for allowing different allocations of friendly (also known as the Blue force) effects against the enemy. The Blue force is created in a similar fashion, but is not given any ordering or allocation instructions – those will be performed in the analysis phase. The end result of the interactive event creation process is an *Event Description*.

Event Description. The tactical event is described by the static Red plan (described above), the Blue force, and the resources available to each. The resources are direct-fire (DF) effects, such as tanks or mechanized infantry, indirect-fire (IF) effects that suppress direct-fire elements, minefield (MF) effects, and mine-clearing (MC) effects. The static Red plan provides the configuration of all of the Red resources, but the Blue resources are initially not allocated. The *Event Description* is passed into the *Analyzer*, which can be either genetic-algorithm-based or crisp-rules-based.

[illegible]

Mechanism. As the allocation is processed each engagement of blue effects against a red force within a level is converted into a force summary. The *Force Summary* is fed into the Results Interface, and the modifications made on the summary by the Combat Results Mechanism are provided back to the Generations Agent as *Results*. For each allocation the Generations Agent creates and maintains a Status data structure that parallels the allocation. This Status structure holds the intermediate results. The next time a Blue effect is used its initial strength is taken from the Status structure, not the original allocation.

Combat Results Mechanism. The Combat Results Mechanism receives a *Force Summary* of Blue and Red effects involved in an engagement and determines the outcome by running it through a results determination process. In this project, either a simple combat results table or the crisp-rules system can be used. However, this could be any sufficiently robust tactical simulation that can provide “good enough” answers in “fast enough” time. The *Results* of the combat are returned to the Generations Agent / Fitness Monitor.

Generations Agent / Fitness Monitor. The Generations Agent / Fitness Monitor keeps track of the best allocations and determines which one to present to the user. When all of the *Results* are returned for a particular allocation a fitness function is applied. This fitness function is configurable, but currently only through code changes. This fitness evaluation is the mechanism for biasing the selection of the “best” allocations. An example fitness function is to select allocations that reduce all Red forces by the same percentage (which might cause lower Blue end strength. Alternatively, the fitness function can select for lowest overall Red end-strength, which might leave an enemy force completely untouched. A simple example fitness function that provides reasonable results is the maximization of Blue direct-fire end-strength modified by the proportion of Red direct-fire strength that survives. In other words, the more surviving Blue strength and the less surviving Red strength, the better.

With probability related to their fitness, two allocations are selected for reproduction. Each pair creates a new pair for the next generation. As two allocations combine to produce two children for the next generation there is a high probability (default of 0.70) that bits of the parent genome will crossover to form the child genome. This crossover is also subject to event constraints mentioned above (indirect-fire can only be used once, etc.). After each child allocation has been created, there is a very low probability (default of 0.01) of mutation of any gene, again subject to constraints from the event.

Most-Fit Allocation. The Survival Test identifies the “best” outcome based on the utility function. Once the best outcome of the event (*Most-Fit Allocation*) is determined it is passed back to the GUI for display to the user.

Analyzer: Crisp-Rules-Based

The crisp-rules-based analyzer is very flexible. It can be used at the lowest level as a fundamental results mechanism. It can be used to evaluate all engagements within one level. It can

also be used to evaluate all engagements within different levels, providing the complete result for the entire event. The crisp-rules-based analyzer is based on the Java Expert System Shell (JESS), an expert system shell and scripting language written in the Java language (Friedman-Hill 1999). JESS is based on the CLIPS rule-based system. JESS supports the development of rule-based expert systems, and these rules can be tightly coupled to code written in the Java language. Although CLIPS and JESS have a capability for handling fuzzy concepts and reasoning, the rules used in this project are crisp and do not have a probabilistic component in determining which rule to fire.

The rules-based agent has a total of 22 rules and there are three levels of salience. Salience allows a priority-based scheme to be placed within the rules. In order of priority they are the default level, the step change level, and the phase change level. The rules are in a separate file and are loaded at run time. They can easily be modified or updated without changing the program code.

The rule-based system goes through a series of steps to resolve a single event or battle. Forces are allocated, then combat is resolved, and the results are evaluated for success. If not successful then a new allocation will be tried until either all forces are expended unsuccessfully or a force mix is found that is successful. The rules follow the same combat model that the GA analyzer uses in resolving combat effects.

Rules are also in place to allow the rule-based system to step through the list of events that need to be achieved. Once all events have either been successfully or unsuccessfully tried the rule-based system tabulates the results and passes them back to the GUI for display.

The rule-based agent tries to use a minimum of force to achieve the objectives. This means that other options are currently not tried. However, expanding on the rules present in the system can modify additional biases.

The advantage of the rule-based system is that it can present a view of its computations that is understandable by a human operator. This is not as simple with the multiple generations of the GA approach. The intermediate results are output as text messages while the rule-based system moves through its rules in computing a solution.

Visualization of the Event Resolution

The *Most-fit allocation* from the GA-based analyzer or the results from the crisp-rules-based analyzer (including intermediate results) are provided back to the GUI. The allocation of effects is applied to the graphical entities and the user can “walk through” the levels to see how the allocation occurred. Simultaneously, the strength results are extracted from the Status structure and used to modify the strength display for each entity. See Figure 2 for an example of an event resolution on the system screen, and Table 1 for the corresponding allocation.

The Red outpost force in level 0, to the west, was subject to the entire strength of the Blue elements, and has been destroyed

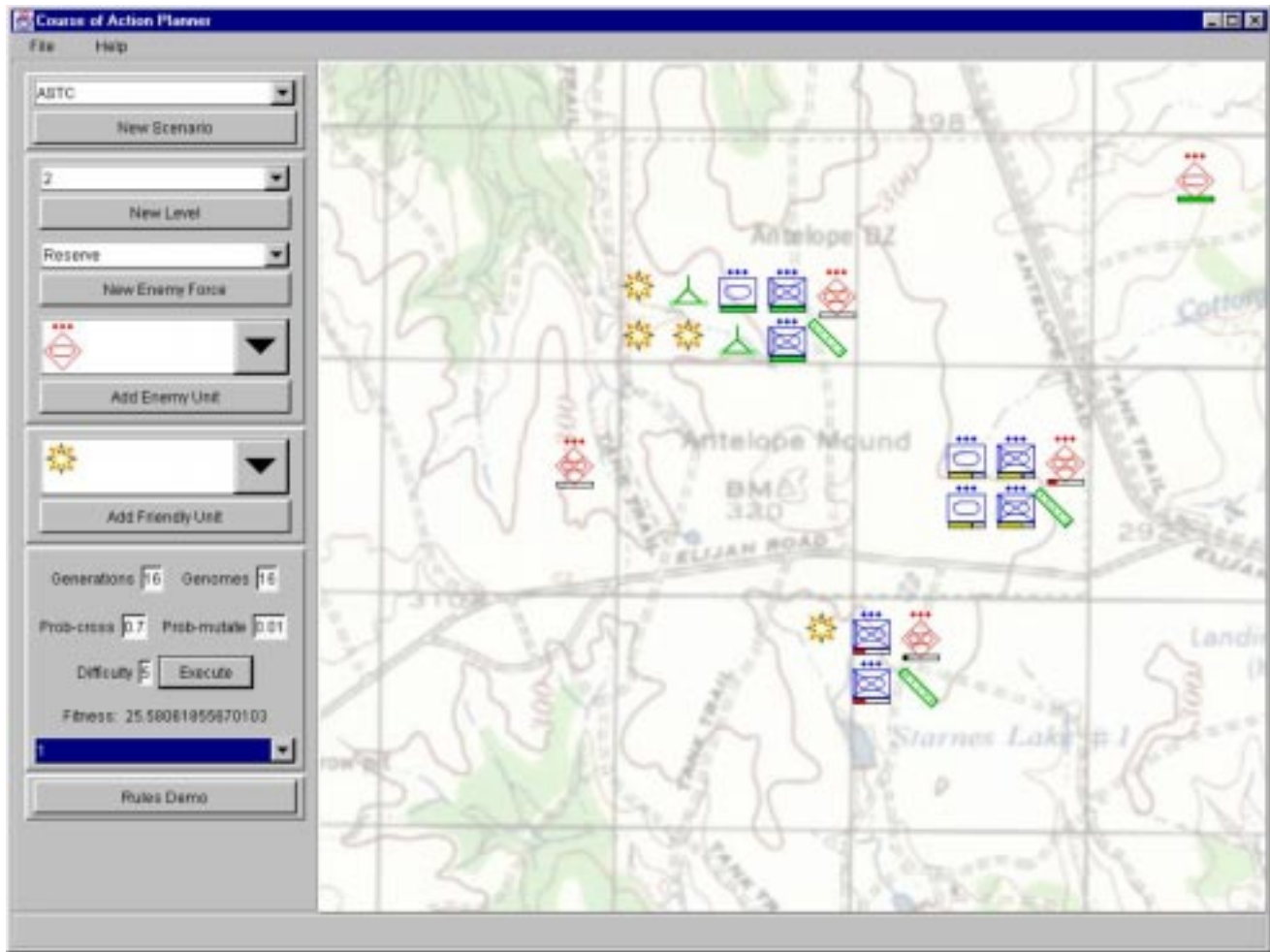


Figure 2: Example Visualization of Tactical Event Resolution – Second Level

and over-run. The screen is displaying the results of the second level, which represents a main defensive belt. The Red force to the north consists of a mechanized infantry effect (M) with a minefield effect (MF). It has been allocated two Blue mech effects, one tank (T) effect, two mine-clearing (MC) effects, and three indirect fire (IF) effects. The Red force in the center (one mech, one minefield) has been allocated two Blue mech effects and two tank effects. The Red force to the south (one mech, one minefield) has been allocated two Blue mech effects and one Blue indirect fire effect.

At the end of the engagement, the strengths (represented by status bars under the icons) of the three Blue DF effects in the north are “green,” meaning fully combat effective. In the center, all four Blue DF effects are high amber, meaning weakened but effective (amber does not show well in black and white). In the south, the two Blue DF effects are both red, meaning they will not be effective in future engagements. All of the Red effects in this level have been reduced to a status of red or black, meaning completely ineffective. The Red reserve force, to the northeast, will not be engaged until level 2, so it still has a green status.

This level demonstrates the influence of the biased agents. The mobility/counter-mobility agent has applied all available MC assets against a single enemy force, while the fire support agent has chosen to mass indirect fires against the same force. In tactical terms, both the mobility/counter-mobility and fire support agents are supporting the maneuver agent’s choice of an economy-of-force mission in the south, a block task in the center, and a penetration of the enemy defensive belt in the north.

RESULTS

The entire system performed remarkably well, from the GUI interface through the event descriptions into the analyzers and back out to the GUI display. This allowed several tests to be conducted with different parameters. In every test, the outcomes of the analysis of each event seemed quite reasonable, and the evaluation of the project was favorable; however, there are a few limitations of the system that should be noted.

Testing

The test protocol used an incremental approach to evaluate and demonstrate the system. Initially, the tests were limited to resolving a single event with direct-fire (DF) engagements with only a few DF effects on each side, and satisfactory DF engagement outcomes were demonstrated. Indirect fire (IF) and minefield (MF) / mine-clearing (MC) effects were added, and also yielded satisfactory results. From that point on, increasing levels of complexity and numbers of effects were added, all of which yielded good results. When parameters for the GA analyzer were changed for complex scenarios, improvements were noted with increased numbers of genomes over more generations. As the situations became more complex the time consumed by Java object creation was overcome by the development of the *Force Summary*, after which the system ran an order of magnitude faster.

Limitations

There are several limitations to this first prototype system. First, the system does not consider side effects and consequences external to the event that could have an impact on the event. For instance, using artillery in one engagement may cause the guns to be targeted for counter-battery fire, reducing the availability of indirect-fire effects in the next event. Second, the system allows direct-fire forces to conduct unrealistic maneuvers, such as attacking first in the south, then moving all the way up to the north in the next level. Finally, the allocation mechanism in the GA-based analyzer can suffer from a local maximum problem, wherein much better allocations can't be reached.

CONCLUSION

Perhaps the most important question to be asked in evaluating this system is whether the desired capabilities were achieved. This project was very successful in implementing the desired capabilities. Automation of the tactical event resolution process significantly reduced the amount of time required. The GUI of the prototype system made input and ordering of an event into a simple process and provided good visualization of the results. Different ideas from human planners about event resolution, including biases from battlefield functions such as maneuver and fire support can be incorporated into the system through the biased agents mechanism. As the system evaluates the event it keeps track of resource status. The system provides the planner with the ability to enforce an ordering on the resolution of the event, providing the planner with more control. The system has the ability to capture intermediate results and include them in the visualization, helping the planner to understand the resolution of the event. The genetic algorithm and the crisp rules proved to be very effective techniques for examining and selecting allocations. Finally, the combat results mechanism is a separate entity, providing the capability to choose the mechanism providing the appropriate mix of accuracy, speed, and computational efficiency.

REFERENCES

- Army, U. S. 1997. *Field Manual 101-5, Staff Organization and Operations*, U.S. Government Printing Office, Washington, D.C.
- DARPA. 2000. "Command Post of the Future (CPOF) Project." Available at <http://dtsn.darpa.mil/iso/>. Last accessed on January 18th, 2000.
- DeJong, K. A. 1975. "An Analysis of Behavior of a Class of Genetic Adaptive Systems," Ph.D. Dissertation, University of Michigan.
- Forgy, C. L. 1982. "RETE: A Fast Algorithm for the Many Pattern / Many Object Pattern Match Problem." *Artificial Intelligence*, 19(1), 17-37.
- Franklin, S., and A. Graesser. 1997. "Is it an Agent, or Just a Program?: A Taxonomy for Autonomous Agents." Published in *Intelligent Agents III: Proceedings of the Third International Workshop on Agent Theories, Architectures, and Languages*, J. P. Muller, M. J. Wooldridge, and N. R. Jennings, eds., Springer-Verlag, Berlin, 21-35.
- Friedman-Hill, E. 1999. "Java Expert Systems Shell (JESS)." Version 5.0. Available at <http://herzberg.ca.sandia.gov/jess>.
- Giarratano, J., and G. Riley. 1989. *Expert Systems Principles and Programming*, PWS-Kent Publishing Company, Boston.
- Goldberg, D. E. 1989. *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison Wesley, Reading, Massachusetts.
- Hayes, C. C., and J. L. Schlabach. 1998. "FOX-GA: A Planning Support Tool for assisting Military Planners in a Dynamic and Uncertain Environment." Technical Report WS-98-02, AAI, Madison, Wisconsin.
- IDFL. 2000. "Interactive Displays Federated Laboratory." Available at <http://www.ifp.uiuc.edu/IDFL/>. Last accessed on January 19th, 2000.
- Kewley, R. H. 1999. "Automated Tactical Course of Action Development." Operations Research Center, United States Military Academy, West Point, New York.
- Russell, S. J., and P. Norvig. 1995. *Artificial Intelligence: A Modern Approach*, Prentice Hall, Upper Saddle River, New Jersey.
- Schlabach, J. L., C. C. Hayes, and D. E. Goldberg. 1998. "FOX-GA: A Genetic Algorithm for Generating and Analyzing Battlefield Courses of Action." *Evolutionary Computation*, 7(1), 45-68.