Intelligent Agents for the Synthetic Battlefield: A Company of Rotary Wing Aircraft

Randall W. Hill, Jr., Johnny Chen, Jonathan Gratch, Paul Rosenbloom, Milind Tambe

Information Sciences Institute and Computer Science Department
University of Southern California
4676 Admiralty Way
Marina del Rey, CA 90292
{hill, chen, gratch, rosenbloom, tambe}@isi.edu

Abstract[1]

We have constructed a team of intelligent agents that perform the tasks of an attack helicopter company for a synthetic battlefield environment used for running large-scale military exercises. We have used the Soar integrated architecture to develop: (1) pilot agents for a company of helicopters, (2) a command agent that makes decisions and plans for the helicopter company, and (3) an approach to teamwork that enables the pilot agents to coordinate their activities in accomplishing the goals of the company. This case study describes the task domain and architecture of our application, as well as the benefits and lessons learned from applying AI technology to this domain.

Task Description

Background

Since 1983 the Defense Advanced Research Projects Agency (DARPA) has exerted a significant effort to develop a realistic, distributed, synthetic battlefield that could be used for training, mission planning and rehearsal, tactics and doctrine development, and weapon-system concept evaluation. The underlying distributed interactive simulation (DIS) technology builds large-scale simulations from a set of independent simulators linked together in a network (DIS Steering Committee 1994). It is envisioned that a synthetic battlefield will make it cheaper and safer to conduct large scale military exercises than would be possible with live field units. One of the goals of DARPA's Synthetic Theater of War '97 (STOW-97) project is to field several thousand entities during one exercise, using a DIS environment called ModSAF (Modular Semi-Automated Forces)(Calder et al., 1993).

Simulation in DIS is high-fidelity: actions in ModSAF are represented and resolved at the level of individual combatants and vehicles, which we hereafter refer to as entities. The synthetic environment includes terrain, oceans, and environmental effects (e.g., weather), all of which affect the perception and actions of the entities.

It is very expensive to field thousands of human troops for a field exercise, but deploying thousands of ModSAF simulation entities is also a challenge. One of the weaknesses of the synthetic forces in ModSAF is that they are only capable of a limited amount of autonomy. Entities can be tasked to perform low-level actions, e.g., move in formation along a route, and attack an objective, but they are unable to perform a complex mission without a human controller to intervene and guide them from task to task. Furthermore, even with human controllers, the behavior of the ModSAF entities often does not achieve the desired level. Consequently, running a large-scale simulation could potentially be very costly in terms of the number of human controllers that would be required, and the quality of the simulation may not reach the desired level. Herein lies the motivation for applying artificial intelligence techniques to the development of entities in this domain.

This paper describes our effort to address these challenges. This work is part of a two phase project by a consortium of researchers from the University of Michigan (UM)[2], University of Southern California Information Sciences Institute (USC-ISI), and Carnegie Mellon University (CMU). The first phase emphasized the development of individual Soar/IFOR (Intelligent Forces) agents for the fixed wing aircraft domain. During this phase we developed a system using the Soar agent architecture (Tambe et. al, 1995; Laird et. al, 1995). This has served as the foundation for the work described in this paper. The second phase of the project has continued the development of the Soar/IFOR agents, with UM developing new missions and capabilities in the fixed wing aircraft domain and CMU continuing their focus on Natural Language Processing. At USC-ISI we shifted our focus to the helicopter domain, while also adding new techniques to facilitate teamwork (Tambe, 1996a) and command-level decision making (Gratch, 1996). Here we describe how these new approaches help address the challenge of

[2] We wish to acknowledge the UM team for their role with USC-ISI in developing the FWA agent architecture: Karen Coulter, Randy Jones, Frank Koss, John Laird, and Paul Nielsen.

constructing intelligent synthetic forces. We limit the scope of the discussion to the development of army helicopter companies although other entities have also been implemented within this framework.

Overview of the Attack Helicopter Company

The army attack helicopter company (AHC) has eight attack helicopters and pilots, and a company commander who plans the company's missions and makes high-level decisions for the company. There are usually three AHC's in an attack helicopter battalion, and we can deploy all three companies using our application software. The battalion also has a commander, but we do not yet model this agent. Instead, a human represents the battalion commander: the battalion-level plans are generated by a human and sent in an operations order, which is the standard military message for communicating mission plans to subordinates.

In a typical mission the battalion commander sends an operations order to each of the AHC commanders, which are all modeled as Soar/CFOR agents. Each of the AHC commanders analyzes the operations order and plans the mission for their respective companies. Once the planning is complete and the company's plan has been approved, the AHC commander sends an operations order to the AHC pilot agents, who execute the mission. The mission usually involves following a route to a position deep in enemy territory, avoiding contact with the enemy along the way to the battle area, destroying enemy targets in an engagement area, and returning to home base once the mission's objectives have been achieved.

Objectives

Our overall goal is twofold. First, we seek to develop intelligent agents capable of acting autonomously on the synthetic battlefield so as to reduce the amount of human control needed to run large-scale military exercises. Second, we want our agents to be as human-like in behavior as possible so that the quality of the simulation meets the high standards of the domain experts.

To meet these goals, we need to develop AHC pilot agents that can fly their helicopters and execute their missions in the context of a complex, dynamic, and uncertain synthetic-battlefield environment, and they need to be able to act autonomously for relatively long periods of time -- hours to days at a time. In addition, the individual pilot agents within the company must be able to act as a team during the execution of a mission to achieve the company's goals.

With respect to the company command agent, we need to model decision-making from a broader perspective and over longer time scales than what is done by the individual pilot agents. Whereas the pilot agents' decision-making tends to be more reactive in nature, the commander must deliberate about alternate courses of action, project effects into the future, and detect harmful (or beneficial) interactions between subordinate units and enemy forces.

This paper is a case study on how we applied AI technology to the development of the agents in the attack helicopter company. We begin by describing the application's design and implementation, giving an extended example of what tasks the agents in the AHC perform. Next we describe the use of AI technology in our agents, starting with the basic agent architecture and briefly describing the extensions to incorporate teamwork and planning. Following this, we briefly describe how the application is being used, what payoffs are expected, how much effort went into developing it, and what is involved in maintaining it.

Application Description

Design and implementation

The overall architecture of the helicopter company is shown in Figure 1. The distributed interactive simulation environment is provided by ModSAF: each ModSAF application program runs on a different machine on the network. Each ModSAF simulates multiple entities (i.e., vehicles), which are multicast to the other ModSAF's on the network. While ModSAF provides the simulation of the vehicles, in our case the AH-64 Apache helicopter, we implemented the pilot agents who fly the helicopters and the company command agent in Soar (we describe Soar in the section on AI Technology).

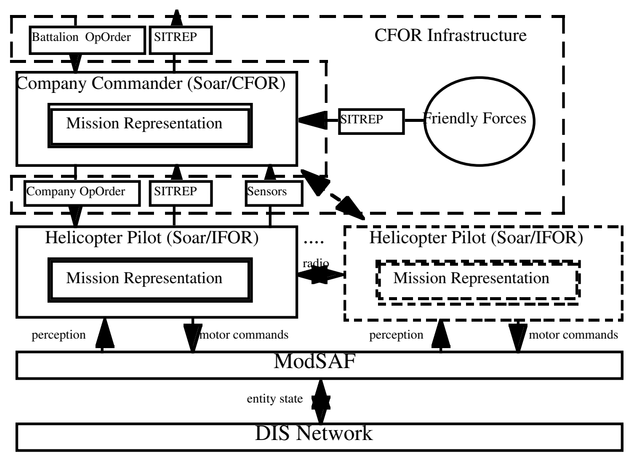

Figure 1: Architecture of the Attack Helicopter Company

The Soar helicopter pilot agents perceive and take actions in the synthetic environment via the Soar-ModSAF interface (SMI), which is a collection of 'C' routines that provide the agents access to the dynamics and state of their vehicles as well as providing simulated sensors for detecting other entities in the environment via vision and radar. The agents control their aircraft, manipulate simulated lasers and weapons, sense the terrain, and communicate by radio via the SMI.

Besides using ModSAF radios to communicate, the Soar helicopter pilot agents and the company commander agent also communicate via the Command and Control Communications Interface Language (CCSIL), (Hartzog and Salisbury, 1996), a structured language that enables the agents to send standard military messages (e.g., operations orders or situation reports) to one another. CCSIL messages are delivered using the Command Forces (CFOR) Infrastructure software (Salisbury, 1995), which uses remote procedure calls to enable agents outside of ModSAF to communicate with agents in ModSAF vehicles. This is important since the Company Commander agent (see Figure 1) does not run in ModSAF but needs to be able to communicate with the Soar helicopter pilot agents that do. In addition, CCSIL provides a convenient way for humans to compose, send, and receive messages from ModSAF and CFOR agents.

The CFOR Infrastructure software also provides an interface to the sensors on ModSAF vehicles as well as providing a set of terrain reasoning functions that can be used by the commander agent in planning routes for tactical movements.

Examples of how system is used

In developing the behaviors of the company, we have focused on the primary mission of an attack helicopter company, the attack mission.[1] In a typical attack mission, the battalion commander, who in this case is a human, issues a plan, called an operations order, to each of the company commanders in the battalion. The operations order contains the battalion's mission, what is known about the current friendly and enemy situations, and specific orders for each of the companies concerning what they are supposed to do as well as when and where they are supposed to do it. For example, the section of the operations order for Company A may state:

On order, ingress on route RED to Holding Area BRAVO, then move to Battle Position CAIRO and attack by fire to destroy enemy forces in Engagement Area HORNET.

Besides telling the companies their missions, the operations order also contains information about the locations of routes, objectives, phase lines, battle positions, engagement areas, and other control measures.

The company commander agent receives and analyzes the operations order, then plans a course of action to achieve the mission. During its initial planning the commander agent takes into consideration the terrain and the enemy situation, and it seeks to find a route to its assigned battle position that maximizes cover and concealment from the enemy force while minimizing the distance traveled. When the command agent has completed its initial planning, it encodes the plan as an operations order and backbriefs it to the battalion commander, who has the option of approving or disapproving it. Once the operations order is approved, it is sent to the individual pilot agents, who read it, identify their tasks and roles in the plan, and begin to execute the mission.

In a typical attack mission, the company organizes itself into two teams, a light team and a heavy team. The light team leads the company formation and performs the role of scouting the route. The heavy team follows the light team and overwatches their progress, providing firepower support when the light team is attacked. The teams take off from the home base and follow a route to a holding area, which is an intermediate point just prior to the company's battle position. Once in the holding area, the helicopters assigned the *scout* role move forward to the battle position, flying nap-of-the-earth to remain hidden from the enemy. The scouts reconnoiter the battle position to determine whether it is safe for the company to move forward and whether there are enemy targets in the engagement area. If enemy forces are observed in the engagement area, the scouts call the company forward to the battle position.

When the company commander agent receives the locations of the enemy vehicles from the scouts, it replans the company's firing positions so as to maximize cover and concealment and to optimize the missile ranges to the enemy vehicles. The company commander sends the new firing positions to the helicopter pilots, who move to and occupy their individual positions. Once the helicopters are in their firing positions, the pilots first mask (hide) behind terrain features such as a hill or ridge to conceal themselves from the enemy. To engage a target, the helicopter unmasks, selects one or more targets, fires several missiles, and quickly re-masks. In order to insure its own survivability, the helicopter will shift its position laterally while masked and unmask in another position.

When the termination criteria for engaging targets has been met, the company re-groups and flies back to the home base. If they should encounter an unexpected enemy force enroute to their objective, it is necessary to take "actions on contact," which involves immediately reacting to the situation in order to insure survival and modifying their current plans to bypass or destroy the enemy. In some cases it is desirable for the company to evade the enemy force and avoid contact altogether. In this instance the commander must re-plan the route to keep the company out of the enemy's weapon range, and, if possible, out of visual contact with the enemy. In other cases it is desirable to engage the enemy force to suppress or destroy it. In this instance, if the company can see the enemy and is out of the enemy's weapons range, the commander needs to re-plan the route so that the company can approach and engage the enemy unit from a position that provides cover and concealment.

[1] Other potential missions include reconnaissance and security.

Use of AI Technology

Soar-based agent architecture

The pilot agents and the commander agent are built within Soar, a software architecture that is being developed as a basis for both an integrated intelligent system and a unified theory of human cognition (Rosenbloom, Laird & Newell, 1993; Newell, 1990). Soar provides the agents with support for knowledge representation, problem solving, reactivity, external interaction, and learning, (though our agents do not currently take advantage of Soar's learning capabilities), and it allows for the smooth integration of planning and reaction in decision-making (Pearson et al., 1993; Laird & Rosenbloom, 1990).

Tasks and goals are represented in Soar as *operators*. Operators perform the deliberate acts of the system. They can perform simple, primitive actions that modify the internal state and/or generate primitive external actions, or they can perform arbitrarily complex actions, such as executing a mission. The basic processing cycle is to repeatedly propose, select, and apply operators to a state. Operator selection, application, and termination are all dynamically determined by the system's knowledge and preferences, stored in the form of productions. Any changes in goals, states, and perceptions can cause these productions to fire.

In our application, agents interact with the DIS environment in real-time. This ability is facilitated by Soar's incorporation of perception within its decision loop. Decisions are informed by the current state of the world, as well as by rules that fire to interpret it. The perception system clusters visual entities based on proximity and summarizes this data in the agent's visual input, making it easier for the agent to reason about groups of individuals.

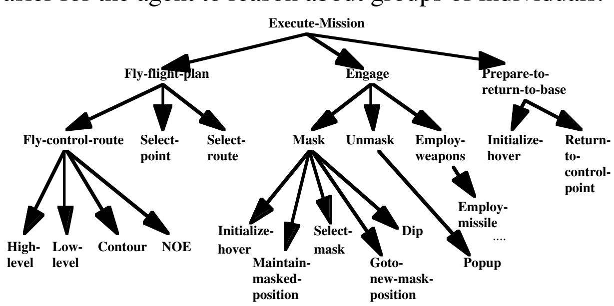

Figure 2: Part of the operator hierarchy for a pilot agent

Figure 2 shows a portion of the operator hierarchy of the basic helicopter agent. The Execute-mission operator decomposes into operators associated with different aspects of the attack mission. For example, the Fly-flight-plan operator decomposes into operators that enable the agent to fly a route from one location to another. Once the attack helicopter company has occupied the battle position, the Engage operator hierarchy is used for performing the tactics to destroy an enemy target while minimizing the danger to oneself. These operators serve to illustrate the kind of knowledge used and how it is organized for the attack helicopter domain.

Teamwork

Teamwork in complex, dynamic domains, such as synthetic battlefields mandates highly flexible coordination and communication to surmount the uncertainties; e.g., dynamic changes in a team's goals, team members' unexpected inabilities to fulfill responsibilities (because they may crash, get shot down, run out of ammunition, or drop off the simulation network), and communication difficulties. Unfortunately, implemented multi-agent systems often rely on pre-planned, domain-specific coordination that fails to provide such flexibility (Jennings, 1995). First, it is difficult to anticipate and pre-plan for all possible coordination failures, given the complexity of the domain. Second, given domain specificity, reusability suffers -- coordination has to be redesigned for each new domain.

A fundamental reason for these teamwork limitations is the current agent architectures. Architectures such as Soar (Tambe et. al, 1995), RAP (Firby, 1987), IRMA (Pollack, 1992), BB1 (Hayes-Roth et. al, 1995), and PRS (Rao et. al, 1993) facilitate an individual agent's flexible behaviors via mechanisms such as commitments and reactive plans. However, flexible individual behaviors, even if simultaneous and coordinated, do not sum up to teamwork. A common example provided is ordinary traffic, which although simultaneous and coordinated by traffic signs, is not teamwork. Indeed, theories of collaboration point to novel mental constructs as underlying teamwork, such as team goals, mutual beliefs and joint commitments (Grosz, 1996; Cohen and Levesque, 1991), absent in current agent architectures. Thus, agents cannot explicitly represent their team goals and plans, or flexibly reason about their communication/coordination responsibilities in teamwork; instead they rely on the (problematic) pre-planned coordination.

In our work, we have integrated a set of teamwork capabilities within Soar; the combined system is called STEAM (Tambe, 1996a; Tambe, 1997). STEAM is founded on the *joint intentions* theory (Cohen and Levesque, 1991). It enables explicit representation of team goals that expand out into goals and plans for individuals' roles in the team goal. In practice, to enable multiple team members to maintain a coherent view of their team's goals and plans, STEAM additionally incorporates (1) team synchronization to establish joint intentions; and (2) monitoring and repair capabilities. Unfortunately, communication in service of coherent teamwork can itself be a significant overhead or risk (e.g., radio silence in synthetic battlefields). Therefore, STEAM also integrates decision theoretic communication selectivity -- in the best interest of the team, agents may selectively avoid communication.

STEAM's capabilities are encoded in 251 general Soar rules that can be reused across domains. (These rules, along with documentation and traces, are available at http://www.isi.edu/soar/tambe/steam/steam.html). STEAM has been reapplied in the context of Marine transport helicopter simulation, where a team of escorts and transports transport synthetic troops from sea to land. STEAM is also in the process of being applied in the RoboCup soccer simulation. Evaluations of STEAM's flexibility and reusability are presented in (Tambe, 1997).

Planning

The demands of command-level decision making require a greater focus on deliberation than required by pilot agents. The commander must be proactive rather than reactive, anticipating the possible outcomes of future actions as well as potential interactions that might arise between actions. The greater focus on deliberation has led us to draw substantially from the classical planning literature in the course of command entity development. The command entity incorporates a hybrid of planning styles, incorporating hierarchical task-decomposition techniques (as in Tate, 1990) as well as partial-order planning approaches (as in Penberthy and Weld, 1992; Gratch, 1996). In this respect it closely resembles the IPEM planning architecture of Ambros-Ingerson and Steel (1988), with some significant enhancements. Task-decomposition planners plan by decomposing abstract tasks into a set of more concrete subtasks. Partial-order planners utilize the causal relationships between tasks to recognize various plan flaws such as missing tasks or ordering conflicts.

In the command agent, plans are represented by a graph structure known as a hierarchical task network (HTN). Nodes in the network correspond to tasks, and are represented as STRIPS-style action descriptions (Fikes, 1971). Tasks may be abstract or primitive. Abstract tasks may be decomposed into a partially ordered set of more specific tasks. Primitive tasks are those that may be directly executed without further decomposition.

Tasks in the network are connected by a variety of relations between them. Subtask relations define the basic hierarchical structure of the network. Ordering relations define the order in which tasks should be executed. Causal links and protection intervals are relations which represent the causal structure of the plan. (As in standard partial-order planners such as UCPOP, this causal structure facilitates reasoning about interactions across tasks.) Finally, dependency relations record information about how the plan was generated, for use in replanning. (Dependency relations are similar to the dependency graphs of Hayes (1975) and the verification structure of Kambhampati (1992).)

The three key activities performed by the command agent are plan generation, plan execution monitoring, and replanning. For plan generation, initially the planner is given an abstract plan (a battalion order). Typically the initial plan cannot be executed. It may contain non-primitive tasks or tasks may have unsatisfied preconditions. The flaws in the initial plan are addressed by non-deterministically selecting planning operations to resolve these flaws: fleshing out an abstract task (decomposition), finding existing tasks that satisfy preconditions (simple establishment), adding tasks (step addition), etc. These planning operations result in modification to the HTN and a complete plan is constructed through backtracking search.

Plan execution monitoring is facilitated through the use of a current world description. This is a variable-free set of literals that represents the sensed state of the world at the current time step. Execution operators allow tasks to be initiated or terminated. Only one task may be initiated or terminated on a give time step, but multiple tasks may be executing simultaneously. A task may be initiated if no unexecuted tasks precedes it and its preconditions unify with the current world description. A task may be terminated if its effects unify with the current world description. Currently, we do not have a general method for handling execution failure and rely on domain-specific procedures.

The current world description also facilitates the recognition of unexpected events. If there is a literal in the current world description that does not unify with any effect of any executed action, it is assumed to be the effect of some external event and is inserted into the task network.

Replanning may be triggered in response to unexpected events. If the effect of an unexpected event creates a flaw in the existing plan, the planner must modify the plan to address this new factor. The plan can be repaired in one of two ways: extending the plan (by adding new constraints or tasks), or retracting some portion of the plan and replanning. The former is the easiest approach, however it is not always possible to retain the original plan structure. If the unexpected event is incompatible with existing plan structure, this structure must be retracted. This process is facilitated through the use of the decision graph. This graph records the dependencies between planning decisions. In a method similar to Hayes' robot planner (1975), the command entity attempts to retract just those planning decisions directly affected by the new state of the world, retaining as much of the old plan as possible.

Application Use and Payoff

Operational Usage

The attack helicopter company (along with Marine escort and transport units) has been deployed in several exercises and tests as a part of the STOW-97 program. It was first deployed in the Fall of 1995 for two events, first for a subsystem integration test in September and then for a major combined engineering demonstration called ED-1 in October. It has been deployed numerous times since then: Combined Test 1, in July, 1996; Combined Test 3, in October, 1996; and Combined Test 4, in December 1996, where we successfully ran three companies simultaneously.

Each of these events has been a test leading up to a major exercise called STOW-97, which will take place in late 1997.

When the AHC has been deployed, thus far, the software was operated by members of our development team. The battalion operations orders, which are the input to the AHC, have been all written by outside subject matter experts (SME's). These operations orders are often provided within hours or days of the exercise, and, in spite of the short preparation time, the commander agent and the pilot agents have generally been able to cope with whatever missions they have been assigned, though we have had to occasionally refine the agents' knowledge base to accommodate new situations.

Benefits of the Application

The payoff from this application is that our AHC has capabilities currently not available to standard ModSAF entities[1]: (1) the commander agent accepts operations orders from its higher echelon and it plans the mission for the company, (2) the Soar/IFOR helicopter pilot agents operate autonomously -- they both execute plans from the commander and react to unexpected contingencies in the environment, (3) the behavior is higher in quality, and (4) the Soar/IFOR agents work as a team in executing their mission.

Taken together, these capabilities should provide a payoff to STOW-97 by reducing the number of simulation operators needed to run a large-scale exercise. Without these capabilities, an operator would have to control multiple individual entities. Instead, the human will play the role of battalion commander and write operations orders and initialize the scenario to control three companies (18-24 helicopters) at a time. We expect the pay-off to increase as we make progress toward developing a battalion command agent so that eventually the human will play the role of the brigade commander or higher while the rest is controlled by intelligent agents.

Development Effort Required

The original Soar/IFOR project focused on the FWA domain, and there is an ongoing development effort to continue this effort (e.g., see Tambe et al., 1995 and Jones et al., 1996). We began working on the Soar/IFOR AHC application in July, 1994. We were able to build on top of a considerable amount of the original Soar/IFOR FWA infrastructure code, which included the interface between Soar and ModSAF. We have invested approximately ten work-years of effort into the development of the AHC, which includes the pilot agents, commander agent, and the supporting infrastructure.

[1] Hughes Research Laboratory and Science Applications International Corporation (SAIC) have also built command entities, but they are have focused on the ground force domain.

Maintenance

Since this application is still being developed, the knowledge base is currently maintained by us, the developers. When we participate in an exercise, which occurs about once every two months, there are subject matter experts (SME's) who observe and critique the behavior of the AHC. The critiques are provided in a written form and changes are subsequently made to the knowledge base to incorporate the suggested changes. In addition to the changes suggested by the SME's, we are still adding new knowledge from task description documents and refining the knowledge as we test the agents. It is expected that the number of changes needed will decrease over time as the behavior is refined and the SME's validate that it is doctrinally correct.

There are several ways that changes to the domain knowledge occur. First, the domain knowledge related to attack helicopter doctrine and tactics should only change when the US Army decides it is necessary to do so (an inherently slow process). So, on the surface, the domain knowledge should be fairly stable. From a practical point of view, however, a second way that domain knowledge changes is when two or more SME's disagree on a fine point of doctrine or tactics. This may lead to a refinement of the domain knowledge that was not previously documented. Finally, there is domain knowledge that we consider to be common sense in nature -- it is not written as tactics and has to do with the way agents reason about things like space, terrain, and teamwork. This type of knowledge is continually being refined as we discover ways in which the agents' behavior differs from a human's.

The process of modifying the knowledge base requires a detailed knowledge of Soar and of the agents' problem space hierarchy. Since the knowledge is organized as a set of problem spaces, updating the knowledge base is made somewhat easier since changes are usually localized to a particular problem space. Alleviating the difficulties of modifying the knowledge base remains an issue for future work.

References

Ambros-Ingerson, J.A. and S. Steel, 1988. Integrating Planning, Execution, and Monitoring. In Proceedings of the National Conference on Artificial Intelligence, 1988, pp. 83-88.

Calder, R.B., J.E. Smith, A.J. Courtemanche, J.M.F. Mar, A.Z. Ceranowicz, 1993. ModSAF Behavior Simulation and Control. In Proceedings of the Second Conference on Computer Generated Forces and Behavioral Representation, STRICOM-DMSO, July, 1993, pp. 347-356.

Cohen, P. R. and Levesque, H. J., 1991. Confirmation and Joint Action. In Proceedings of International Joint Conference on Artificial Intelligence.

DeJong, G, and Mooney, R, 1986. Explanation-based learning: An alternative view. *Machine Learning* 1, 2, 1986, pp. 145-176.

DIS Steering Committee. 1994. The DIS Vision: A Map to the Future of Distributed Simulations. Technical Report, IST-SP-94-01, Institute for Simulation and Training, University of Central Florida.

Erol, K., Hendler, J., and Nau, D.S.., 1994. HTN Planning: complexity and expressivity. In Proceedings of the National Conference on Artificial Intelligence, 1994, pp. 1123-1128.

Fikes, R. E., and Nilsson, N. J., 1971. STRIPS: a new approach to the application of theorem proving to problem solving. *Artificial Intelligence Journal,* 2 (3/4), 1971.

Firby, J., 1987. An investigation into reactive planning in complex domains. In Proceedings of National Conference. on Artificial Intelligence.

Gratch, J.A., 1996. Task-decomposition planning for command decision making. In Proceedings of the Sixth Conference on Computer Generated Forces and Behavioral Representation, STRICOM-DMSO, July, 1996, pp. 37-45.

Grosz, B., 1996. Collaborating systems. *AI Magazine*, vol. 17.

Hartzog, S. M., Salisbury, M.R., 1996. Command Forces (CFOR) Program Status Report. In Proceedings of the Sixth Conference on Computer Generated Forces and Behavioral Representation, Orlando, Florida, July, 1996.

Hayes, P. J., 1975. A representation for robot plans. In Proceedings of the International Joint Conference on Artificial Intelligence, 181-188, 1975.

Hayes-Roth, B., Brownston, L. and Gen, R.V., 1995. Multiagent collaboration in directed improvisation. In Proceedings of International Conference on Multi-Agent Systems.

Jennings, N., 1995. Controlling cooperative problem solving in industrial multi-agent systems using joint intentions. *Artificial Intelligence Journal*, 75, pp. 195-240.

Jones, R.M., Laird, J.E., and Nielsen, P.E., 1996. Moving intelligent automated forces into theater-level scenarios. In Proceedings of the Sixth Conference on Computer Generated Forces and Behavioral Representation, STRICOM-DMSO, July, 1996, pp. 113-117.

Kambhampati, S., 1992. A validation-structure-based theory of plan modification and reuse. *Artificial Intelligence Journal* 55: 193-258.

Laird, J.E., Johnson, W.L., Jones, R.M., Koss, F., Lehman, J.F., Nielsen, P.E., Rosenbloom, P.S., Rubinoff, R., Schwamb, K., Tambe, M., Van Dyke, J., van Lent, M., Wray, R.E., III, 1995. "Simulated Intelligent Forces for Air: The Soar/IFOR Project 1995," Proceedings of the Fifth Conference on Computer Generated Forces and Behavioral Representation, STRICOM-DMSO, pp. 27-36.

Laird, J.E. and Rosenbloom, P.S., 1990. Integrating execution, planning, and learning in Soar for external environments. In Proceedings of the National Conference on Artificial Intelligence. Menlo Park, California: The AAAI Press, July, 1990.

Mitchell, T. M., Keller R. M., and Kedar-Cabelli, S. T., 1986. Explanation-based generalization: A unifying View. *Machine Learning* 1 (1): 1986, pp. 47-80.

Newell, A., 1990. *Unified Theories of Cognition*. Harvard University Press, Cambridge, Massachusetts, 1990.

Pearson, D.J., Huffman, S.B., Willis, M.B., Laird, J.E., and Jones, R.M., 1993. A symbolic solution to intelligent real-time control. *IEEE Robotics and Autonomous Systems*, 11:279-291.

Penberthy, J., Weld, D., 1992. UCPOP: A sound, complete, partial order planner for ADL. In Proceedings of the 3rd International Conference on Principles of Knowledge Representation and Reasoning, 103-114.

Pollack, M., 1992. The uses of plans. *Artificial Intelligence Journal*, volume 57.

Rao, A. S., Lucas, A., Morley, D. Selvestrel, M. and Murray, G., 1993. Agent-oriented architecture for air-combat simulation. Technical Report 42, Australian AI Institute.

Rosenbloom, P.S., Laird, J.E., Newell, A., (Eds.) 1993. *The Soar Papers: Research on Integrated Intelligence*. Cambridge, MA: MIT Press.

Salisbury, M.R., Booker, L.B., Seidel, D.W., Dahmann, J.S., 1995. Implementation of command forces (CFOR) simulation. In Proceedings of the Fifth Conference on Computer Generated Forces and Behavioral Representation, 423-430, Orlando, Florida, May, 1995.

Tambe, M., Johnson, W.L., Jones, R.M., Koss, F., Laird, J.E., Rosenbloom, P.S., and Schwamb, K., 1995. Intelligent agents for interactive simulation environments. *AI Magazine*, 16(1):15-39, Spring, 1995, AAAI Press.

Tambe, M., 1996a. Teamwork in real-world, dynamic environments. In Proceedings of the International Conference on Multi-Agent Systems.

Tambe, M., 1996b. Tracking dynamic team activity. In Proceedings of the National Conference on Artificial Intelligence.

Tambe, M., 1997. Agent architectures for flexible, practical teamwork. In Proceedings of the National Conference on Artificial Intelligence.

Tate, A., 1990. Generating project networks. In Allen, J.; Hendler, J; and Tate, A., editors, Readings in Planning. Morgan Kaufman. 162-170.