The Computer Journal Advance Access published May 12, 2009

doi:10.1093/comjnl/bxp038

Analyzing Team Decision-Making in Tactical Scenarios

GITA SUKTHANKAR[1,*] AND KATIA SYCARA[2]

[1]*School of Electrical Engineering and Computer Science, University of Central Florida, Orlando FL 32816-2362, USA*
[2]*Robotics Institute, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh PA 15213, USA*
[*]*Corresponding author: gitars@eecs.ucf.edu*

Team decision-making is a bundle of interdependent activities that involve gathering, interpreting and exchanging information; creating and identifying alternative courses of action; choosing among alternatives by integrating the often different perspectives of team members and implementing a choice and monitoring its consequences. To accomplish joint tasks, human team members often assume distinctive roles in task completion. We believe that to design and build software agents that can assist human teams, we need develop automated techniques to identify the roles of the human decision-makers. If the supporting agents are insensitive to shifts in the team's roles, they cannot effectively monitor the team's activities. This article addresses the problem of doing offline role analysis of battle scenarios from multi-player team games. The ability to identify team roles from observations is important for a wide range of applications including automated commentary generation, game coaching and opponent modeling. We define a *role* as a preference model over possible actions based on the game state. This article explores two promising approaches for automated role analysis: (1) a model-based system for combining evidence from observed events using the Dempster–Shafer theory and (2) a data-driven discriminative classifier using support vector machines.

Keywords: pattern recognition; teamwork; multi-player games; evidential reasoning

Received 3 October 2008; revised 17 January 2009
Handling editor: Erol Gelenbe

1. INTRODUCTION

A team is a form of organizational structure in which its members have compatible goals and decision-making in a bundle of interdependent activities that involve gathering, interpreting and exchanging information; generating alternative courses of action; choosing among alternatives by integrating the often different perspectives of team members; and implementing a choice and monitoring its consequences. Coalition forces are one example of team decision-making in time-pressured, high-stakes situations. Providing automated decision support for such environments is a very challenging problem, due to shortening decision cycles, the changing nature of threats, opponent tactics and environmental unpredictability. Intelligent agents have the promise to provide timely assistance in various areas of decentralized, collaborative decision-making, such as information gathering, information dissemination, monitoring of team progress and alerting the team to various unexpected events. In order to fulfill the promise of agent technology in providing effective team assistance, developing automated techniques to analyze human teamwork is crucial. The goal of our research project is to develop automated analysis techniques to allow for effective agent support for human teams that are engaged in adaptive teamwork in dynamic environments.

This article addresses the problem of analyzing multi-player tactical battle scenarios from game logs. The ability to identify individual and team plans from observations is important for a wide range of applications including constructing opponent models, automated commentary, coaching applications and surveillance systems. However, the military adage 'no plan, no matter how well conceived, survives contact with the enemy intact'[1] reveals that in many cases a team's plan execution halts

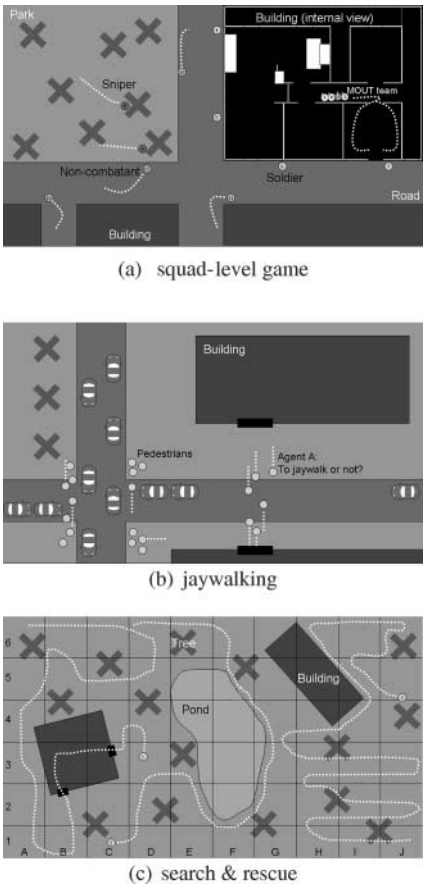
[1]This quote has been attributed to numerous military figures but originally stems from Prussian field marshal Helmuth von Moltke the Elder's statement 'Therefore no plan of operations extends with any certainty beyond first contact with the main hostile force.' [1]

early in the course of battle due to unexpected enemy actions. Of course, an ideal plan would include courses of action for all possible contingencies, but typical battle plans only include options for a small set of expected outcomes. If the enemy's actions cause the world state to deviate from this expected set, the team is often forced to abandon the plan. After multiple plans have been initiated and abandoned, matching the observation trace becomes a difficult proposition, even for an omniscient observer. Moreover, it is unclear whether expert human teams create deep plan trees in situations where enemy actions may force plan abandonment after a few time steps.

Even in cases when the pre-battle plan has been abandoned, we hypothesize that successful teams continue to follow a *policy* through the course of battle and that this policy can be recovered from the observed data. We define a *policy* as a preference model over possible actions, based on the current game state. A *team policy* is a collection of individual policies along with an assignment of players to policies (roles). Policies are typically broad but shallow, covering all possible game states without extending far through time, whereas *plans* are deep recipes for goal completion, extending many time steps into the future, but narrow, lacking contingencies for all but a small set of expected outcomes. The same player intentions can be expressed as either a plan or a policy for game play.

In this article, we present two techniques for recovering individual and team policies from multi-player tactical scenarios. Our scenarios are described and played using the Open Gaming Foundation d20 Game system (v3.5) [2]. The d20 system is a set of rules governing combat, negotiation and resource management employed by popular turn-based tabletop games.

The remainder of the article is organized as follows. Section 2 discusses decision-making within human groups. Section 3 summarizes related work on goal, policy and plan recognition in games. Section 4 defines the policy recognition problem, describes the d20 rule system and presents the battle scenarios that are used by our human players. The game logs from these battles provided the data on which our policy recognition approaches are evaluated. Sections 5 and 6 present two complementary methods for policy recognition: an evidential reasoning system for scoring data from game logs and a discriminative classifier trained on simulated game logs. Section 7 discusses results in the context of plan recognition and Section 8 concludes the article.



(a) squad-level game

(b) jaywalking

(c) search & rescue

FIGURE 1. Team behavior interactions: (a) actions of agents in cohesive teams are tightly coupled in both space and time; (b) agents in emergent groups are significantly influenced by their social context; and (c) agents assume roles to effectively cover the region.

2. GROUP INFLUENCE ON HUMAN BEHAVIOR

To appreciate the range of ways in which groups can influence human behavior, consider the three scenarios shown in Fig. 1.

Figure 1a shows an overhead view of a squad-based game where several agents operating in cohesive units attempt to achieve a specified objective (e.g. storming an enemy position). A team may be spatially clustered, such as the squad exploring the building or distributed, such as the group guarding the perimeter of that building. In either case, team members execute rehearsed plans such as bounding overwatch (for advancing along a street) or the buttonhook maneuver (for entering a room). This degree of tightly coupled coordination is possible because the agents share goals, beliefs and plans and can typically communicate extensively before (or often during) the scenario. The challenges in analyzing this scenario include the following: (1) given that multiple squads are simultaneously operating on the field, assign each observed agent to a squad; (2) identify the current behavior and plan state of each squad; and (3) correctly handle the dynamic merging and splitting of teams. In earlier work we have proposed methods to efficiently address this problem using simultaneous team assignment and behavior recognition [3]. Our approach exploits the significant spatial and temporal constraints on the observable states of the agents imposed by squad-level maneuvers.

The second scenario, presented in Fig. 1b, focuses on the actions of a solitary pedestrian agent (labeled 'A') who must decide whether to risk jaywalking in order to efficiently cross

the street. Studies of human behavior have shown [4,5] that jaywalking is significantly influenced to do so on the behavior of other pedestrians in the environment. For example, a tourist may be more likely to jaywalk if he sees a group of well-dressed businessmen jaywalking but might be less influenced to do so on seeing a single teenager run across the street. Similarly, a crowd of people waiting impatiently for a slow pedestrian crossing frequently jaywalk *en masse* once a few individuals trigger the event. Conversely, the presence of high-status individuals (and authority figures such as policemen) who refrain from jaywalking can inhibit the incidence of jaywalking. In other words, although our pedestrian agent is not formally in a team, its behavior is significantly influenced by the actions of other agents in its environment; although the influence of other agents in the environment is subtler than in the previous scenario.

The third scenario, shown in Fig. 1c, presents a cooperative search-and-rescue (or scavenger hunt) task being performed by a loosely coordinated set of agents. As in the first scenario, the agents share a common objective but they operate in a more independent manner. This task requires that the agents coordinate their coverage patterns and plan their search paths so as to efficiently cover the space while minimizing unnecessary overlap. Task performance is correlated with effective role allocation to team members. In comparison with the first scenario, the spatial and temporal constraints between agent positions are much weaker since teams do not typically employ pre-determined formations. Although agent actions may not have strong spatial or temporal constraints, they can still be analyzed in terms of the *role* that an agent serves for the team.

3. RELATED WORK

In this section, we present a brief overview of related work on analyzing player actions in games and military scenarios. Single-player keyhole plan recognition was implemented for text-based computer adventure games by Albrecht *et al.* [6] where dynamic Bayesian networks (DBNs) were used to recognize quest goals and to predict future player actions. Mott *et al.* [7] demonstrated a similar goal recognition system for interactive narrative environments using scalable n-gram models. Unlike those systems, our work focuses on the tactical aspect of battlefield adventures where multi-player interactions, limited player knowledge and stochastic action outcomes significantly increase the degree of unpredictability.

Behavior recognition has also been explored in the context of first-person shooter computer games. Moon *et al.* [8] analyzed team effectiveness in America's army game in terms of communication and movement patterns. Rather than recognizing behaviors, their goal was to distinguish between effective and ineffective patterns. Sukthankar and Sycara [9] employed a variant of hidden Markov models to analyze military team behaviors in an Unreal Tournament based exclusively on the relative physical positioning of agents.

Team behavior recognition in dynamic sport domains has been attempted using both model-based and data-driven approaches. Intille and Bobick [10] developed a framework for recognizing known football plays from multi-agent belief networks that were constructed from temporal structure descriptions of global behavior. Bhandari *et al.* [11] applied unsupervised data mining and knowledge discovery techniques to recognize patterns in NBA basketball data. Recently, Beetz *et al.* [12] developed a system for matching soccer ball motions to different action models using decision trees. The work on behavior recognition in sports has focused primarily on the mapping of movement traces to low-level game actions (e.g. scoring and passing). In contrast, our article examines sequences of higher-level agent actions and game state to infer the player's policy and current tactical role in the team.

Policy recognition has been applied to problems in the Robocup domain. Chernova and Veloso [13] presented a technique to learn an opponent evasion policy from demonstration. Kuhlmann *et al.* [14] fitted a team's movement patterns to a parametric model of agent behavior for a coaching task. Patterns were scored according to their similarity to models learned from the pre-game logs. This work is conceptually similar to our data-driven approach for policy recognition.

4. DOMAIN

To simulate the process of enemy engagement, we adapted the combat section of Open Gaming Foundation's d20 System (v3.5) to create a set of multi-player tactical scenarios. The d20 system has several useful properties that make it a promising domain:

(i) The d20 game system is turn-based rather than real-time. A game can thus be logged as a sequence of discrete actions performed by each player and policy recognition can be directly executed on these streams of actions and observed game states.

(ii) The outcome of actions is stochastic, governed by rolling dice (typically an icosahedral die, abbreviated as *d20*). The rules define the difficulty levels for a broad range of combat tasks; to determine whether a particular action succeeds, the player rolls a d20 and attempts to score a number that is greater than or equal to the difficulty level of the task.

(iii) Spatial arrangements affect the outcome of many of the combat actions, making the difficulty level easier or harder; for instance, two allies on opposite sides of a target gain a mutual benefit from *flanking* the enemy. Thus, the tactical arrangements of units on the grid can significantly influence the course of a battle. Experienced players coordinate such actions to maximize their chance of success.

(iv) Teamwork between players is of paramount importance. The opposing forces are designed to be impossible for a single player to defeat. However, certain game rewards are occasionally awarded to the first player to achieve an objective. To succeed, players must simultaneously contribute to team goals in battle while pursuing their own competitive objectives.

A typical d20 Game is played by a group of 3–6 players with one referee. Each player controls one character within the virtual gaming world that the referee brings to life through verbal descriptions and miniatures on a dry-erase tabletop gaming map. Characters have a well-defined set of capabilities that determine their competence at various tasks in the virtual world. The referee controls the actions of all other entities in the virtual world (known as *non-player characters*). During a typical four-hour session, the referee poses a series of challenges—diplomatic negotiations, battles and puzzles—that the players must cooperatively solve. Success for the players enhances the capabilities of their characters, giving them more abilities and resources. Failure can result in the death of characters or the loss of resources Thus, characters' capabilities persist over multiple gaming sessions, which makes it different from other iterative games where the state is reset at the conclusion of each session.

Although tabletop gaming lacks the audio and visual special effects of its increasingly popular computerized cousins, tabletop games have a stronger emphasis on team tactics and character optimization. In battle, time becomes a limited resource and players typically have to coordinate to overcome their foes before the foes defeat them. Since most computer games are real time rather than turn based, they disproportionally reward fast keyboard reflexes and manual dexterity over tactical and strategic battle planning. For instance, D&D Online, a real-time, multi-player game based on the d20 gaming system, requires the players to accurately move and shoot during combat, whereas the tabletop version used in our study only requires players to select their location and choose an action.

4.1. Multi-player tactical scenarios

Our experiments in policy recognition focus on the subset of the d20 system actions that deal with tactical battles (as opposed to diplomatic negotiation or interpersonal interaction). Each scenario features a single battle between a group of player characters and a set of referee-controlled opponents. The players have limited knowledge of the world state and limited knowledge about their opponents' capabilities and current status; the referee has complete knowledge of the entire game state. Players are allowed to communicate in order to facilitate coordination but the referee will usually disallow excessive communication during battle and limit players to short verbal utterances.

Figure 2 shows a typical multi-player tactical scenario. The three players must cooperate to achieve one of two objectives:

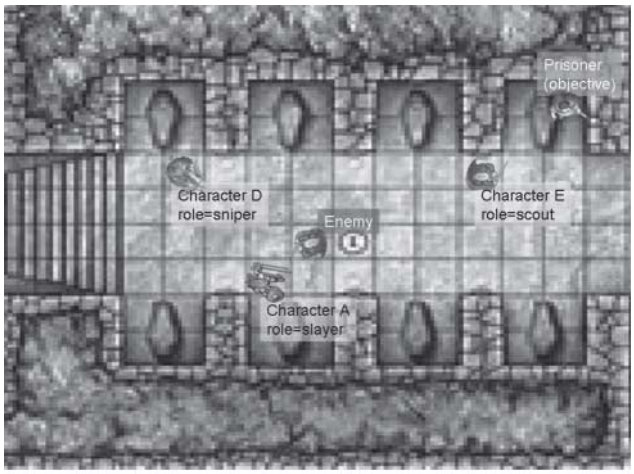


FIGURE 2. (Primary tactical scenario). The three human players select roles that enable them to achieve one of two goals: (1) defeat the opponent in battle; and (2) distract the opponent while one character frees the prisoner.

TABLE 1. Characters and summarized capabilities.

Name	Offensive	Defensive	Special	Stealth
A	high	medium	low	low
B	medium	high	low	low
C	low	medium	medium	low
D	high	low	medium	medium
E	medium	low	low	high
F	medium	low	high	medium

defeat their opponent or distract the opponent to rescue the prisoner in the corner of the room. Based on the capabilities of their characters, the players select a goal and allocate functional roles for the upcoming battle. Each of the three players was assigned a character; the capabilities of the characters are summarized in Table 1. Each character is capable of fulfilling multiple roles in a team but each is poorly suited to at least one role. The roles are:

(i) *slayer*: who fights opponents in close-combat;
(ii) *blocker*: a defensive character who shields more vulnerable characters;
(iii) *sniper*: who fights at range; and
(iv) *medic*: who restores health to other characters;
(v) *scout*: a stealthy character who can rescue the prisoner without being noticed by the distracted opponent.

4.2. Game mechanics

The battle sequence in each scenario is as follows:

(i) The referee draws the terrain and places the opponents on the grid at the beginning of the scenario. Then the

players place their miniatures on the grid to indicate their starting location.

(ii) At the start of the battle, each entity (players and opponents) rolls an initiative to determine the order of action each round (time unit). This initiative is maintained through the battle (with a few possible exceptions).

(iii) Each round is a complete cycle through the initiatives in which each entity can move and take an action from a set of standard actions.

(v) If a character's health total goes below 0, it is dead or dying and can no longer take any actions in the battle unless revived by another player's actions.

Each character has the following attributes that affect combat actions: (1) *hit points*: current health total; (2) *armor class*: a measure of how difficult a character is to hit with physical attacks; and (3) *attack bonus*: a measure of how capable a character is at handling weaponry. When a character attacks an opponent, the attack is resolved using the following formula to determine whether the attack is successful:

$$d(20) + \text{attack bonus} + \text{modifiers} \geq \text{armor class} + \text{modifiers},$$

where $d(20)$ is the outcome of a twenty-sided die roll. If the expression is true, the attack succeeds and the opponent's hit points are reduced. Situational modifiers include the effect of the spatial positioning of the characters. For instance, the defender's armor class improves if the attacker is partially occluded by an object that provides cover.

4.3. Problem formulation

We define the problem of policy recognition in this domain as follows. Given a sequence of input observations O (including observable game state and player actions) and a set of player policies $\mathcal{P}$ and team policies T, the goal is to identify the policies $\rho \in \mathcal{P}$ and $\tau \in \mathcal{T}$ that were employed during the scenario. A player policy is an individual's preference model for available actions given a particular game state. For instance, in the scenario shown in Fig. 2, the archer's policy might be to preferentially shoot from a distance rather than engaging in close combat with a sword, but he/she might do the latter to protect a fallen teammate. In a team situation, these individual policies can be used to describe a player's role in the team (e.g. combat medic). A team policy is an allocation of players to these tactical roles and is typically arranged prior to the scenario as a locker-room agreement [15]. However, circumstances during the battle (such as the elimination of a teammate or unexpected enemy reinforcements) can frequently force players to take actions that were *a priori* lower in their individual preference model.

In particular, one difference between policy recognition in a tactical battle and typical plan recognition is that agents rarely have the luxury of performing a pre-planned series of actions in the face of an enemy threat. This means that methods that rely on temporal structure, such as DBNs and hidden Markov models are not necessarily be well suited to this task. An additional challenge is that, over the course of a single scenario, one only observes a small fraction of the possible game states, which makes policy learning difficult. Similarly, some games involve situations where the goal has failed and the most common actions for a policy are in fact rarely observed (e.g. an enemy creates a smokescreen early in the battle, forcing the archer to pursue lower-ranked options). The following sections present two complementary approaches to policy recognition: (1) a model-based method for combining evidence from observed events using the Dempster–Shafer theory and (2) a data-driven discriminative classifier using support vector machines (SVMs).

5. MODEL-BASED POLICY RECOGNITION

The model-based method assigns evidence from observed game events to sets of hypothesized policies. These beliefs are aggregated using the Dempster–Shafer theory of evidential reasoning [16]. The primary benefit of this approach is that the model generalizes easily to different initial starting states (scenario goals, agent capabilities, number and composition of the team).

5.1. Dempster–Shafer theory

This section presents a brief overview of the Dempster–Shafer theory of evidential reasoning [16]. Unlike traditional probability theory where evidence is associated with mutually exclusive outcomes, the Dempster–Shafer theory quantifies belief over *sets* of events. The three key notions of the Dempster–Shafer theory are: (1) basic probability assignment functions (m), (2) belief functions (Bel) and (3) plausibility functions (Pl). We describe these below.

The basic probability assignment function assigns a number between 0 and 1 to every combination of outcomes (the power set). Intuitively this represents the belief allocated to this subset and to no smaller subset. For example, after observing an agent's actions over some time, one may assert that it is following either policy ρ_1 or ρ_2, without further committing belief as to which of the two is more likely. More formally, given a finite set of outcomes Θ whose power set is denoted by 2^Θ, the basic probability assignment function, $m : 2^\Theta \mapsto [0, 1]$ satisfies:

$$m(\emptyset) = 0$$
$$\sum_{A \subseteq \Theta} m(A) = 1.$$

Following Shafer [16], the quantity $m(A)$ measures the belief committed *exactly* to the subset A, not the total belief committed to A. To obtain the measure of the total belief committed to A, one must also include the belief assigned to all proper subsets

of A. Thus, we define the belief function $\text{Bel} : 2^{\Theta} \mapsto [0, 1]$ as

$$\text{Bel}(A) = \sum_{B \subseteq A} m(B).$$

Intuitively the belief function quantifies the evidence that directly supports a subset of outcomes. The non-informative belief function (initial state for our system) is obtained by setting:

$$m(\Theta) = 1 \quad \text{and} \quad m(A) = 0 \ \ \forall A \neq \Theta.$$

The plausibility function quantifies the evidence that does not directly contradict the outcomes of interest. We define the plausibility function $\text{Pl} : 2^{\Theta} \mapsto [0, 1]$ as

$$\text{Pl}(A) = \sum_{B \cap A \neq \emptyset} m(B).$$

The precise probability of an event is lower-bounded by its belief and upper-bounded by its plausibility functions, respectively.

We employ the Dempster–Shafer theory to model how observed evidence affects our beliefs about a character's current policy. For instance, seeing a character moving on the battlefield could indicate that the agent's role is that of a *sniper*, a *medic* or a *scout* (rather than a *slayer* or *blocker*). This can be expressed as:

$$m(\{\text{sniper,medic,scout}\}) = 0.7 \quad \text{and} \quad m(\Theta) = 0.3.$$

The *belief* that the agent is adopting one of these roles is 0.7, yet the belief that the agent is specifically a sniper is 0 (although the *plausibility* for either of these is 1). Conversely, while the *belief* that the agent is adopting a slayer policy is also 0, the plausibility is only 0.3.

The Dempster–Shafer theory also prescribes how multiple, independent sources of evidence should be combined. Dempster's rule of combination [16] is a generalization of Bayes' rule and aggregates two basic probability assignments m_1 and m_2 using the following formula:

$$m_{12}(\emptyset) = 0,$$
$$m_{12}(C \neq \emptyset) = \frac{\sum_{A \cap B = C} m_1(A) m_2(B)}{1 - \sum_{A \cap B = \emptyset} m_1(A) m_2(B)}.$$

One potential issue with this rule is its treatment of conflicting evidence. The normalizing term in the denominator redistributes the probability mass associated with conflicting evidence among the surviving hypotheses. Under certain conditions, this has the undesirable property of generating counterintuitive beliefs. In the pathological case, an outcome judged as unlikely by both m_1 and m_2 can have a value of 1 in m_{12} if all other subsets conflict. To address this problem, several other rules of combination have been proposed [17].

Yager's rule [18] is very similar to Dempster's rule with the single exception that conflicting evidence is assigned to the universal set, $m(\Theta)$, rather than used as a normalizer. The rule can be stated as follows:

$$m_{12}(\emptyset) = 0,$$
$$m_{12}(C \neq \emptyset) = \sum_{A \cap B = C} m_1(A) m_2(B),$$
$$m_{12}(\Theta) \leftarrow m_{12}(\Theta) + \sum_{A \cap B = \emptyset} m_1(A) m_2(B).$$

Although this formulation is not associative, we implement Yager's rule in a quasi-associative manner to enable efficient online update [17]. In the absence of conflict, Yager's rule gives the same results as Dempster's rule of combination.

Finally, we consider another quasi-associative rule: the intuitive idea of averaging corresponding basic probability assignments [17]:

$$m_{1\ldots n}(A) = \frac{1}{n} \sum_{i=1}^{n} m_i(A).$$

Unfortunately it is impossible to know *a priori* which rule will perform well in a given domain since no single rule has all the desirable properties.

Evidential reasoning has yielded good results on sensor fusion tasks (see [19] for a survey), especially in cases where the sensors produce highly conflicting evidence. Although we do not observe the team members with physical sensors, our technique must be able to cope with conflicting evidence 'sensed' from the observed game events.

5.2. Empirical evaluation

Based on domain knowledge, we identified a general set of eight observable events that occur during a simulated battle. Events were chosen to be (1) externally observable, (2) executable by multiple classes of characters, and (3) related to combat rather than diplomacy or logistics. Each of these events was associated with a basic probability assignment function to assign beliefs over sets of individual policies. For example, the observed event of a character being attacked by an opponent is associated with: $m(\Theta) = 0.1$, $m(\{\overline{\text{scout}}\}) = 0.4$ and $m(\{\text{blocker}\}) = 0.5$. This rule assigns a large belief (0.5) to the blocker policy, while reducing the plausibility of the scout policy to 0.1. Note that the set $\{\overline{\text{scout}}\}$ also includes the *blocker* policy, and thus the plausibility of *blocker* is 1. The plausibility of the remaining three policies is 0.5. The m-functions for the set of events observed for each character was aggregated using the three rules of combination described in Section 5.1.

The events included in the sets were derived from the authors' expert knowledge of the d20 system; doing an exhaustive search and evaluating the performance of every set combination is intractable. However, we believe that it would be possible to tune the system by starting with a large initial event set assignment and performing feature selection.

TABLE 2. Confusion matrix for model-based policy recognition.

	Dempster (%)					Yager (%)					Averaging (%)				
	Sniper	Slayer	Medic	Blocker	Scout	Sniper	Slayer	Medic	Blocker	Scout	Sniper	Slayer	Medic	Blocker	Scout
Sniper	100	0	0	0	0	100	0	0	0	0	88	0	0	12	0
Slayer	0	83	0	17	0	0	83	0	17	0	7	40	7	39	7
Medic	0	0	100	0	0	0	0	100	0	0	0	0	0	100	0
Blocker	0	0	0	100	0	0	0	0	100	0	0	0	0	100	0
Scout	0	0	0	0	100	0	0	0	0	100	0	0	0	0	100

Data from a series of games using the tactical scenario shown in Fig. 2 was recorded and annotated according to our list of observable events. We collected data from five teams with three players on each team; all of the players were expert tabletop gamers who were experienced with the d20 rule system.

We computed the average accuracy over the set of battles for each of the three rules of combination. At the conclusion of each battle, the system made a forced choice, for each player, among the set of policies (roles). Each player was classified into the singleton policy with the highest belief. Comparing this against the ground truth and averaging over battles produces the confusion matrix given in Table 2. We note that, according to this forced-choice metric, all of the combination rules perform reasonably well, with Dempster's rule scoring the best. The largest source of confusion is that the *slayer* policy is occasionally misclassified as *blocker*. This motivates the data-driven method described in Section 6.1 where we specifically learn classifiers to discriminate between these two similar policies.[2]

To illustrate how belief changes as evidence is aggregated using the different combination rules, we plot the belief for each policy for one battle from our data set (Fig. 3). Since neither Yager nor averaging employ normalization, we plot their beliefs on a semi-log scale. We note the following. Yager's rule makes conflicting evidence explicit by allocating significant mass to the *unknown* policy. In particular, this reveals the difficulty of distinguishing between the *blocker* and *slayer*, even late in the battle. A concern with Yager's rule is that the belief in a policy decays over time, despite increasing evidence; this occurs because all rules leak mass to the *unknown* set. Averaging corresponding m-values performs the least well in our domain.

6. DATA-DRIVEN POLICY RECOGNITION

To discriminate between similar policies, we propose a data-driven classification method that is trained using simulated battle data. By training the method on a specific scenario, it can exploit subtle statistical differences between the observed outcomes of similar policies. For instance, characters following the *slayer* policy should both inflict more damage on their opponents and receive more damage in return, whereas the more defensive *blocker* policy occupies the enemy without resulting in substantial losses on either side. This section describes the classifier that we employ, SVM, and evaluates the method on a second scenario (Fig. 4).

6.1. Support vector machines

The goal of policy classification is to label an observed action sequence as a member of one of k categories (e.g. *blocker* vs. *slayer*). We perform this classification using SVMs [20]. SVMs are supervised binary classifiers that have been demonstrated to perform well on a variety of pattern classification tasks. Intuitively the SVM projects data points into a higher-dimensional space, specified by a kernel function, and computes a maximum-margin hyperplane decision surface that separates the two classes. Support vectors are those data points that lie closest to this decision surface; if these data points were removed from the training data, the decision surface would change. Given a labeled training set $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_l, y_l)\}$, where $\mathbf{x}_i \in \Re^N$ is a feature vector and $y_i \in \{-1, +1\}$ is its binary class label, an SVM requires solving the following optimization problem:

$$\min_{\mathbf{w},b,\xi} \frac{1}{2}\mathbf{w}^{\mathrm{T}}\mathbf{w} + C\sum_{i=1}^{l}\xi_i$$

constrained by:

$$y_i(\mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}_i) + \mathbf{b}) \geq 1 - \xi_i,$$
$$\xi_i \geq 0.$$

The function $\phi(\cdot)$ that maps data points into the higher-dimensional space is not explicitly represented; rather, a *kernel* function, $K(\mathbf{x}_i, \mathbf{x}_j) \equiv \phi(\mathbf{x}_i)\phi(\mathbf{x}_j)$, is used to implicitly specify this mapping. In our application, we use the popular radial basis function (RBF) kernel:

$$K(x_i, x_j) = \exp(-\gamma||x_i - x_j||^2), \quad \gamma > 0.$$

Many efficient implementations of SVMs are publicly available; we use LIBSVM [21] because it includes good

[2]The *blocker* and *slayer* policies can generate very similar observable states since intelligent opponents often target the *slayer* preferentially in order to eliminate their biggest threat.

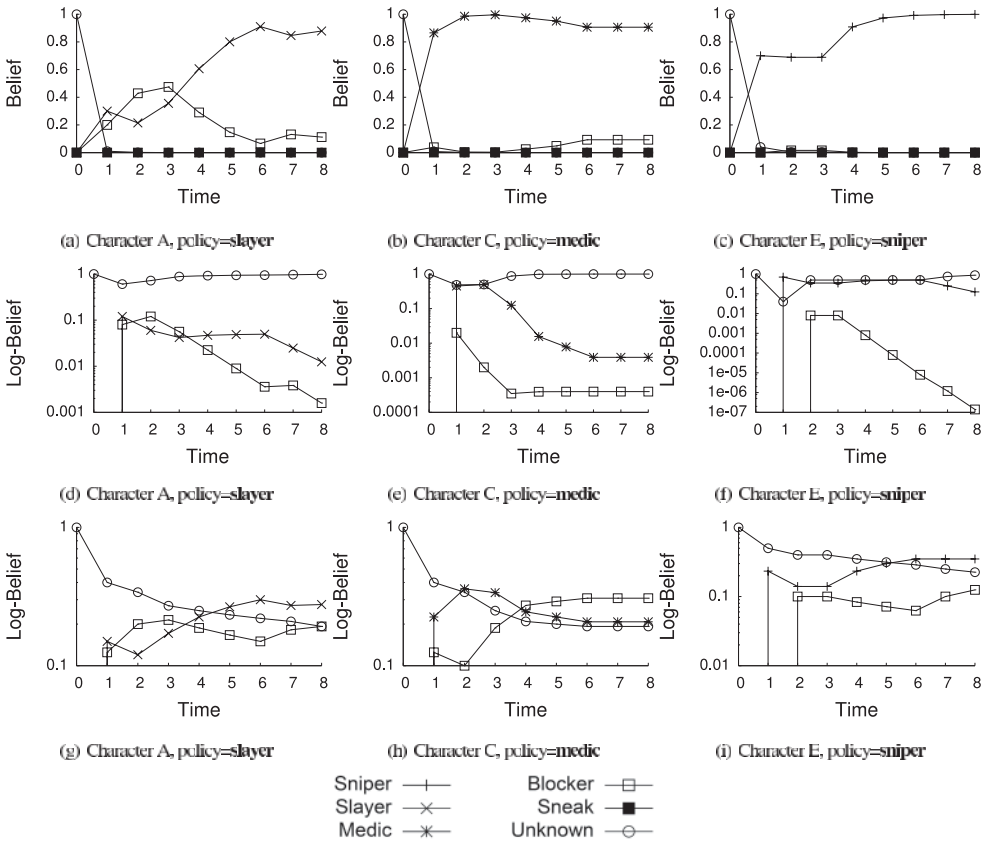


FIGURE 3. Evolution of beliefs over the course of one battle from the scenario shown in Fig. 2. The beliefs for policies adopted by each of the three characters, according to the three rules of combination, are shown. The three rows correspond to Dempster's rule, Yager's rule and m-function averaging, respectively. The columns correspond to three characters A, C and E from Table 1, adopting the ground-truth policies of *slayer*, *medic* and *sniper*, respectively.

routines for automatic data scaling and model selection (appropriate choice of C and γ using cross-validation). To use SVMs for k-class classification, we train $k(k-1)/2$ pair-wise binary classifiers and assign the most popular label.

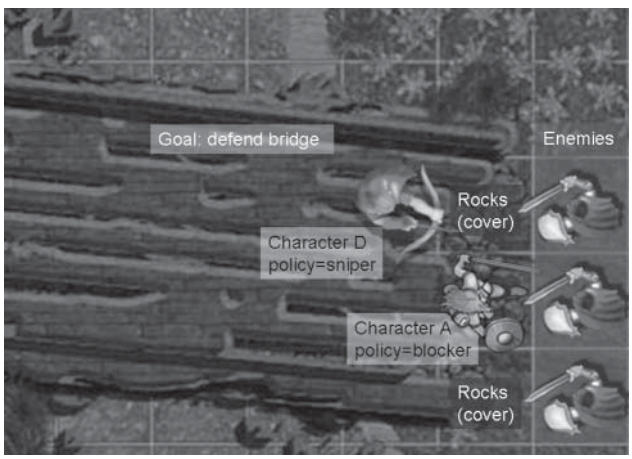


FIGURE 4. Tactical scenario where two players defend a bridge against multiple opponents.

6.2. Empirical evaluation

The data-driven method takes as input a feature vector summarizing the observed information about the battle and performs a forced classification into policies. We investigated three choices for feature sets; and (1) a histogram of the observed character actions over the battle—this is similar to a bag of words model in information retrieval; (2) a vector with character and enemy status at every time step; and (3) a concatenation of these two vectors.

To train the SVM, we generated training data by simulating a set of single player battles in a simplified scenario, using the policies of interest (*blocker* and *slayer*). The trained SVM was then evaluated on several other scenarios. Table 3 shows the confusion matrices for battles on one of these scenarios. This scenario, shown in Fig. 4 is a two-player battle where the characters defend a bridge against multiple opponents; the goal is to correctly classify the policy employed by the front-line character. We observe that the data-driven method using the combined set of features can reliably discriminate between the *blocker* and *slayer* policies. The other two matrices indicate that, in this case, the classifier relies mainly on the histogram

TABLE 3. Confusion matrix for data-driven policy recognition on the scenario shown in Fig. 4 using the three different feature sets.

	Action (%)		State (%)		Combined (%)	
	Blocker	Slayer	Blocker	Slayer	Blocker	Slayer
Blocker	96	4	82	18	98	2
Slayer	0	100	16	84	0	100

of observed actions. However, we note that such data-driven methods require sufficient quantities of training data to avoid overfitting, and that they generalize poorly to novel scenarios when such data have not been provided.

7. DISCUSSION

One interesting aspect about battles in the d20 system is that a player's action choices are constrained by a complex interaction between the character's capabilities, its location on the map and its equipment (including consumable resources). Hence, different players have different attack options, and each player has different choices over time, depending on the current state of the battle. Despite the stochastic nature of the domain, players typically follow battle tactics that are identifiable to other humans. As the human referee controlling the opponents recognizes the players' tactics, he/she will often intelligently adapt to the players' tactics better than computer game engines that are relatively insensitive to player actions. An application of our work would be the development of computerized opponents that react realistically to player tactics to enhance both computer games and military simulations. The d20 tabletop system exercises many of the tactical concepts that current military commanders employ in decision-making games and situational awareness exercises [22].

Teamwork is an important aspect of tactics in the d20 system since the actions of other players can significantly affect the difficulty level of various combat actions. Although building a fully specified team plan is typically impractical given the complexity of the domain, players generally enter battle with a 'locker-room agreement' specifying the policy that the character will seek to use in the upcoming confrontation. There is no simple mapping between a character's capabilities and this policy; an effective team role must consider the capabilities of team members, the expected abilities of opponents and the overall team goal. We believe that identifying each character's policy is an important first step towards predicting the character's future actions, identifying the set of team goals and generating an automated higher-level commentary on the scenario.

The model-based and data-driven approaches are very complementary approaches to battle analysis. The model-based approach generalizes well to other sets of characters, different opponent types and variations in scenario. The data-driven classifier is able to detect subtle statistical differences in action and game state sequences to correctly classify externally similar policies. The data-driven approach does not generalize as well to different character capabilities since a character's capabilities are implicitly incorporated into the training set; thus a testing set that is statistically different cannot be classified accurately. We believe that the two approaches should be combined into a hybrid system where the model-based recognizer identifies high-belief policy sets and the data-driven classifier discriminates between those specific policies. Such an approach is similar in spirit to Carberry's work on incorporating the Dempster–Shafer beliefs to focus heuristics for plan recognition [23].

8. CONCLUSION

This article explores two promising approaches for policy recognition: (1) a model-based system for combining evidence from observed events using the Dempster–Shafer theory and (2) a data-driven classification using SVMs. Evaluation of our techniques on logs of real and simulated games demonstrate that we can recognize player policies with a high degree of accuracy.

Using our game logging methodology and domain-generated *m*-functions, the model-based approach performs extremely well over a broad range of initial conditions. Dempster's rule slightly outperforms the other two rules on the forced classification task. The majority of errors involve confusion between the *blocker* and *slayer* policies, which appear similar at a coarse level. To address this issue, we trained a set of discriminative classifiers using simulated battle logs and evaluated the effects of different feature vectors. The resulting classifiers are highly accurate at classifying these policies, although they do not generalize to characters with different capabilities. Thus, our two approaches are complementary and could be combined into a hybrid policy recognition system to provide detailed automated battle commentary of multi-player tactical scenarios. Our initial implementations, running on a desktop computer, return results in less than 300 ms, making them suitable for interactive online applications, which will be the focus of our future work.

Developing agents that intelligently cohabit in simulated environments with humans will require effective activity/plan recognition. Creating shared understanding between human and agent teammates is the biggest challenge facing developers of mixed-initiative human/agent organizations. The limiting factor in most human–agent interactions is the user's ability and willingness to spend time communicating with the agent in a manner that both humans and agents understand, rather than the agent's computational power and bandwidth [24]. Embedding activity recognition algorithms into multi-player virtual environments will enable:

- the development of smarter opponents that can recognize and respond to the tactics of human teams;

- the creation of better synthetic training partners that can predict their teammates' intentions and coordinate without unnecessary communication;
- automated annotation of multi-player game logs; and
- intelligent user correction in team training environments.

FUNDING

This work has been supported by AFOSR grant F49620-01-1-0542. This research is continuing by participation in the International Technology Alliance sponsored by the U.S. Army Research Laboratory and the U.K. Ministry of Defence under Agreement Number W911NF-06-3-0001.

ACKNOWLEDGMENTS

The authors would like to thank Rahul Sukthankar for his research suggestions.

REFERENCES

[1] Helmuth von Moltke the Elder. http://en.wikipedia.org/wiki/Helmuth_von_Moltke_the_Elder (accessed May 5, 2009).

[2] Open Gaming Foundation d20 Game System (2003) *d20 v3.5 system reference document*. http://www.wizards.com/default.asp?x=d20/article/srd35 (accessed May 5, 2009).

[3] Sukthankar, G. and Sycara, K. (2006) Simultaneous Team Assignment and Behavior Recognition from Spatio-Temporal Agent Traces. *Proc. National Conf. Artificial Intelligence*, Boston, July. AAAI Press.

[4] Mullen, B., Copper, C. and Driskell, J. (1990) Jaywalking as a function of model behavior. *Pers. Soc. Psychol. Bull.*, **16**, 320–330.

[5] Jason, L. and Liotta, R. (1982) Pedestrian jaywalking under facilitating and nonfacilitation conditions. *J. Appl. Behav. Anal.*, **15**, 469–473.

[6] Albrecht, A., Zukerman, I. and Nicholson, A. (1998) Bayesian models for keyhole plan recognition in an adventure game. *User Model. User-Adapt. Interact.*, **9**, 5–47.

[7] Mott, B., Lee, S., and Lester, J. (2006) Probabilistic Goal Recognition in Interactive Narrative Environments. *Proc. National Conf. Artificial Intelligence*, Boston, July. AAAI Press.

[8] Moon, I.-C., Carley, K., Schneider, M., and Shigiltchoff, O. (2005) Detailed Analysis of Team Movement and Communication Affecting Team Performance in the America's Army Game. Technical Report CMU-ISRI-TR-04-100, Carnegie Mellon University, Pittsburgh.

[9] Sukthankar, G. and Sycara, K. (2006) Robust Recognition of Physical Team Behaviors Using Spatio-temporal Models. *Proc. Int. Conf. Autonomous Agents and Multi-agent Systems*, Hakodate, May. IEEE Computer Society.

[10] Intille, S. and Bobick, A. (1999) A Framework for Recognizing Multi-Agent Action from Visual Evidence. *Proc. National Conf. Artificial Intelligence*, Orlando, July. AAAI Press.

[11] Bhandari, I., Colet, E., Parker, J., Pines, Z., Pratap, R. and Ramanujam, K. (1997) Advanced scout: data mining and knowledge discovery in NBA data. *Data Min. Knowledge Discov.*, **1**, 121–125.

[12] Beetz, M., Bandouch, J., Gedili, S., v. Hoyningen-Huene, N., Kirchlechner, B. and Maldonado, A. (2006) Camera-based Observation of Football Games for Analyzing Multi-agent activities. *Proc. Int. Conf. Autonomous Agents and Multi-agent Systems*, Hakodate, May. IEEE Computer Society.

[13] Chernova, S. and Veloso, M. (2006). Tree-based policy learning in continuous domains through teaching by demonstration. In Kaminka, G., Pynadath, D., and Geib, C. (eds), *Modeling Others from Observations: papers from the AAAI Workshop*, 24–31. Technical Report WS-06-13. American Association for Artificial Intelligence, Menlo Park, California, 2006.

[14] Kuhlmann, G., Knox, W. and Stone, P. (2006) Know Thine Enemy: A Champion RoboCup Coach Agent. *Proc. National Conf. Artificial Intelligence*, Boston, July. AAAI Press.

[15] Stone, P. and Veloso, M. (1999) Task decomposition, dynamic role assignment, and low-bandwidth communication for real-time strategic teamwork. *Artif. Intell.*, **110**, 241–273.

[16] Shafer, G. (1976) *A Mathematical Theory of Evidence*. Princeton University Press, Princeton.

[17] Sentz, K. and Ferson, S. (2002) Combination of Evidence in Dempster-Shafer Theory. Technical Report SAND2002-0835, Sandia National Labs, Albuquerque.

[18] Yager, R. (2000) On the Dempster–Shafer framework and new combination rules. *Inf. Sci.*, **41**, 93–137.

[19] Hall, D. and Llinas, J. (1997) An introduction to multisensor data fusion. *Proc. IEEE*, **85**, 6–23.

[20] Vapnik, V. (1998) *Statistical Learning Theory*. Wiley New York.

[21] Chang, C.-C. and Lin, C.-J. (2001). *LIBSVM: a library for support vector machines*. http://www.csie.ntu.edu.tw/∼cjlin/libsvm (accessed May 5, 2009).

[22] Phillips, J., McCloskey, M., McDermott, P., Wiggins, S. and Battaglia, D. (2001) Decision-centered MOUT Training for Small Unit Leaders. Technical Report 1776, U.S. Army Research Institute for Behavioral and Social Sciences, Arlington.

[23] Carberry, S. (1990) Incorporating Default Inferences into Plan Recognition. *Proc. National Conf. Artificial Intelligence*, Boston, July. AAAI Press.

[24] Sycara, K. and Lewis, M. (2004) Integrating Agents into Human Teams. In Salas, E. and Fiore, S. (eds), *Team Cognition: Understanding the Factors that Drive Process and Performance*. American Psychological Association, Washington, DC.