An Agent Architecture for Prognostic Reasoning Assistance

Jean Oh and **Felipe Meneguzzi** and **Katia Sycara**
Robotics Institute
Carnegie Mellon University
Pittsburgh, PA, USA
{jeanoh,meneguzz,katia}@cs.cmu.edu

Timothy J. Norman
Dept. of Computing Science
University of Aberdeen
Aberdeen, UK
t.j.norman@abdn.ac.uk

Abstract

In this paper we describe a software assistant agent that can proactively assist human users situated in a time-constrained environment to perform normative reasoning–reasoning about prohibitions and obligations–so that the user can focus on her planning objectives. In order to provide proactive assistance, the agent must be able to 1) recognize the user's planned activities, 2) reason about potential needs of assistance associated with those predicted activities, and 3) plan to provide appropriate assistance suitable for newly identified user needs. To address these specific requirements, we develop an agent architecture that integrates user intention recognition, normative reasoning over a user's intention, and planning, execution and replanning for assistive actions. This paper presents the agent architecture and discusses practical applications of this approach.

1 Introduction

Human planners dealing with multiple objectives in a complex environment are subjected to a high level of cognitive workload, which can severely impair the quality of the plans created. For example, military planners during peacekeeping operations must plan to achieve their own unit's objectives while following standing policies (or norms) that regulate how interaction and collaboration with Non-Governmental Organizations (NGOs) must take place. As the planners are cognitively overloaded with mission-specific objectives, such normative stipulations hinder their ability to plan to both accomplish goals and abide by the norms. We develop an assistant agent that takes a proactive stance in assisting cognitively overloaded human users by providing prognostic reasoning support. In this paper, we specifically aim to assist in *normative reasoning*–reasoning about prohibitions and obligations.

In order to provide a user with a timely support, it is crucial that the agent recognizes the user's needs in advance so that the agent can work in parallel with the user to ensure the assistance is ready by the time the user actually needs it. This desideratum imposes several technical challenges to: 1) *recognize* the user's planned activities, 2) *reason* about potential needs of assistance for those predicted activities to comply with norms as much as possible, and 3) *plan* to provide appropriate assistance suitable for newly identified user needs.

Our approach to tackle these challenges is realized in a proactive planning agent framework. As opposed to planning for a *given* task, the key challenge we address here is to *identify* a new set of tasks for the agent–*i.e.*, the agent needs to figure out when and what it can do for the user. After identifying new goals, the agent plans, executes, and replans a series of actions to accomplish them. Specifically, we employ a probabilistic plan recognition technique to predict a user's plan for her future activities. The agent then evaluates the predicted user plan to detect any potential norm violations, generating a set of new tasks for the agent to prevent the occurrence of such norm violations. As the user's environment changes the agent's prediction is continuously updated, and thus agent's plan to accomplish its goals must be frequently revised during execution. To enable a full cycle of autonomy, we present an agent architecture that seamlessly integrates techniques for plan recognition; normative reasoning over a user's plan; and planning, execution and replanning for assistive actions.

The main contributions of this paper are the following. We present a principled agent architecture for prognostic reasoning assistance by integrating probabilistic plan recognition with reasoning about norm compliance. We develop the notion of prognostic normative reasoning to predict the user's likely normative violations, allowing the agent to plan and take remedial actions before the violations actually occur. To the best of our knowledge, our approach is the first that manages norms in a proactive and autonomous manner. Our framework supports interleaved planning and execution for the assistant agent to adaptively revise its plans during execution, taking time constraints into consideration to ensure timely support to prevent violations. For a proof of concept experiment, our approach has been fully implemented in the context of a military peacekeeping scenario.

The rest of this paper is organized as follows. After reviewing related work in Section 2, we describe a high-level architecture of our agent system in Section 3. The three main components are described in detail in the following sections: Section 4 describes the agent's plan recognition algorithm for predicting the user's future plan; Section 5 describes how the agent evaluates the norms to maintain a normative state and to detect potential violations; and Section 6 presents how the agent plans and executes actions to accomplish identified

goals. We present a fully implemented system in a peacekeeping problem domain, followed by other potential applications in Section 7, and conclude the paper in Section 8.

2 Related Work

Assistant agents are commonly modeled as planning agents [Boger *et al.*, 2005; Fern *et al.*, 2007] providing maximal help in every possible agent belief state. While these approaches combine both a user's and an agent's variables to define a state space, the models include only the agent's actions. Without an explicit model of the user's actions, these approaches lack the capability to predict user actions, making them unsuitable for prognostic assistance. By contrast, we take a modularized approach. While our plan recognition module keeps track of a user's state variables to predict her current and future activities, the agent's planning problem is defined using only those variables whose values the agent can directly modify. By separating the plan prediction from the agent's action selection, our approach not only achieves an exponential reduction in the size of state space, but also enables the agent to simultaneously assist the user with multiple tasks handled in independent threads.

The type of prognostic assistance described in this paper could be compared to mixed-initiative planning [Burstein and McDermott, 1996] in a certain sense, since the user is planning with the help of the agent. However, since the agent is working only on a subset of the domain in which the user is planning, and we do not expect the user to interact with the agent planning process directly, our approach is distinct from mixed-initiative planning.

As norms are generally complex and dynamically changing, automated norm reasoning is desired in practice. Existing work on automated norm management relies on a deterministic view of the planning model [Modgil *et al.*, 2009], where norms are specified in terms of classical logic; in this approach, violations are detected only after they have occurred, consequently assistance can only be provided after the user has already committed actions that caused the violation [Sycara *et al.*, 2010]. By contrast, our agent aims to predict potential future violations and proactively take actions to help prevent the user from violating the norms.

3 Agent Architecture

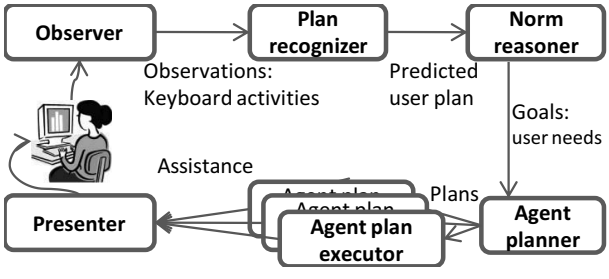

Figure 1: Overview of the proactive assistant.

Figure 1 provides a high-level overview of our agent system for proactive yet unobtrusive assistance. The observer module monitors the user's current activities in the environment to identify new observations that might indicate any changes in the user's current and future plan. Given a new observation, the plan recognizer module uses a probabilistic algorithm to update its prediction for the user's plans. From the predicted user plan, the norm reasoner module evaluates each plan step (actually the state resulting from these steps) to detect any potential norm violations. For each state in which norms are violated, the reasoner module finds the nearest compliant state where violations do not occur. This compliant state subsequently becomes a new goal for the agent to accomplish. The agent planner module then receives the new planning task to find a series of actions to prevent potential norm violations. When a prognostic plan is generated, the agent executes the plan until either the goal is reached or the goal becomes irrelevant to the user's planning context. The effects of the successful plan execution can be presented to the user, *e.g.*, notifying which actions the agent has taken to resolve a certain violation.

Design Assumptions: The agent models a user's planning space in terms of a set of task-specific *variables* and their *domains* of valid values, where a *variable* describes an environment and the progress status of certain activities. A set of norms specifies which states must be visited (or avoided) in the user plan using a set of variables and their relationships. In general, such norms introduce additional variables to consider in addition to task-specific ones, adding extra dimensions into the reasoning process. As seen in a recent study [Sycara *et al.*, 2010], when planning involves complex reasoning as in military environments, human users tend to lose track of norms, resulting in plans with significant norm violations. By developing an assistant agent that manages norm-related variables, our approach aims to relieve the user from having to deal with both task-specific variables and norm-related variables. We make a specific assumption that task-specific *user variables* and norm-specific *agent variables* are independent and thus changing an agent variable does not affect the values of user variables. For representation, let $((\textit{user-variables}), (\textit{agent-variables}))$ denote a state composed of user variables and agent variables.

Example Scenario: We use a simple example of peacekeeping scenario to illustrate the approach throughout the paper. We develop an assistant agent for a humanitarian NGO teamed with a military coalition partner. Consider a norm stating that an NGO must have an armed escort when operating in conflict areas. An escort can be arranged through a well-defined communication protocol, *e.g.*, sending an escort request to and receiving a confirmation from a military party. Here, a state space can be defined in terms of two variables: $area$ specifying the user's geographic coordinates and $escort$ indicating the status of an armed escort in each region. In our approach, a user can focus on reasoning about variable $area$ only since the agent manages variable $escort$ to assure that the user plan complies with norms. Note that variable $escort$ is a simplified representation as it is defined for each value of variable $area$, *i.e.*, it is a function $escort(area)$ to be precise.

In the following sections, we detail the three main components within the agent, namely: plan recognizer, norm reasoner, and agent planner and executor. Due to space con-

straints, we omit the description for the observer and presenter modules.

4 Probabilistic Plan Recognition

We leverage plan recognition from previous work [Oh *et al.*, 2011] and assume that a user's planning problem is given as an MDP. Based on the assumption that a human user generally reasons about consequences and makes decisions to maximize her long-term rewards, we utilize an optimal stochastic policy of the MDP to predict a user's future activities.

We compute an optimal stochastic policy as follows. Let G denote a set of possible goal states. For each potential goal $g \in G$, we compute policy π_g to achieve goal g. Instead of a deterministic policy that specifies only the optimal action, we compute a stochastic policy such that probability $p(a|s,g)$ of taking action a given state s when pursuing goal g is proportional to its long-term expected value $v(s,a,g)$ such that:

$$p(a|s,g) \propto \beta\, v(s,a,g),$$

where β is a normalizing constant. The intuition for using a stochastic policy is to allow the agent to explore multiple likely plan paths in parallel, relaxing the assumption that a human user always acts to maximize her expected reward.

The plan recognition algorithm is a two-step process. In the first step, the algorithm estimates a probability distribution over a set of possible goals. We use a Bayesian approach that assigns a probability mass to each goal according to how well a series of observed user actions is matched with the optimal plan toward the goal. We assume that the agent can observe a user's current state and action. This assumption can be relaxed such that a sequence of user states and actions can be derived from primitive observations, but the detail is omitted here. Let $O_t = s_1, a_1, s_2, a_2, ..., s_t, a_t$ denote a sequence of observed states and actions from time steps 1 through t where s_t and a_t denote the user state and action, respectively, at time step t. When a new observation is made, the assistant agent updates, for each goal g in G, the conditional probability $p(g|O_t)$ that the user is pursuing goal g given the sequence of observations O_t. The conditional probability $p(g|O_t)$ can be rewritten using Bayes' rule as:

$$p(g|O_t) = \frac{p(s_1, a_1, ..., s_t, a_t|g)p(g)}{\sum_{g' \in G} p(s_1, a_1, ..., s_t, a_t|g')p(g')}. \quad (1)$$

By applying the chain rule, we can write the conditional probability of observing the sequence of states and actions given a goal as:

$$\begin{aligned} p(s_1, a_1, ..., s_t, a_t|g) &= p(s_1|g)p(a_1|s_1,g)p(s_2|s_1,a_1,g) \\ &... \quad p(s_t|s_{t-1}, a_{t-1}, ..., g). \end{aligned}$$

We replace the probability $p(a|s,g)$ with the user's stochastic policy $\pi_g(s,a)$ for selecting action a from state s given goal g. By the MDP problem definition, the state transition probability is independent of the goals. Due to the Markov assumption, the state transition probability depends only on the current state, and the user's action selection on the current state and the specific goal. By using these conditional independence relationships, we get:

$$\begin{aligned} p(s_1, a_1, ..., s_t, a_t|g) &= p(s_1)\pi_g(s_1,a_1)p(s_2|s_1,a_1) \\ &... \quad p(s_t|s_{t-1}, a_{t-1}). \end{aligned} \quad (2)$$

By combining Equations 1 and 2, the conditional probability of a goal given a series of observations can be obtained.

In the second step, we *sample* likely user actions in the current state according to a stochastic policy of each goal weighted by the conditional probability from the previous step. Subsequently, the next states after taking each action are sampled using the MDP's state transition function. From the sampled next states, user actions are recursively sampled, generating a tree of user actions known here as a *plan-tree*. The algorithm prunes the nodes with probabilities below some threshold. A node in a plan-tree can be represented in a tuple $\langle t, s, l \rangle$ representing the depth of node (*i.e.*, the number of time steps away from the current state), a predicted user state, and an estimated probability of the state visited by the user, respectively. Example 1 shows a segment of plan-tree indicating that the user is likely be in area 16 with probability .8 or in area 15 with probability .17 at time step t_1.

Example 1 $\langle\langle t_1, (\texttt{area} = 16), .8\rangle, \langle t_1, (\texttt{area} = 15), .17\rangle\rangle$

5 Norm Reasoner

In this section we specify the component responsible for evaluating, using normative reasoning, predicted user plans to generate new goals for the agent. Norms generally define constraints that should be followed by the members in a society at particular points in time to ensure certain system-wide properties. We specify our norm representation format, followed by two algorithms for 1) predicting violations and 2) finding the nearest complying state–*i.e.*the agent's new goal state–towards which we can steer the user.

5.1 Norm Representation

Inspired by the representation in [García-Camino *et al.*, 2009], we define a norm in terms of its deontic modality, a formula specifying when the norm is relevant to a state (which we call the *context condition*), and a formula specifying the constraints imposed on an agent when the norm is relevant (which we call the *normative condition*). We restrict the deontic modalities to those of *obligations* (denoted **O**) and *prohibitions* (denoted **F**); and use these modalities to specify, respectively, whether the normative condition must be true or false in a relevant state. The conditions used in a norm are specified in terms of state variables and their relationships such as an equality constraint. Formally,

Definition 1 (Norm) *A norm is a tuple $\langle \nu, \alpha, \mu \rangle$ where ν is the deontic modality; α, the context condition; and μ, the normative condition.*

Definition 2 (Satisfaction) *Let φ be the set of state variables; α, a context (or normative) condition containing m variables $\varphi_\alpha \subseteq \varphi$ and their valid domain D of m-tuples. We say that condition α is* satisfied *in state s (written $s \models \alpha$) if there exists a tuple in the valid domain that is consistent with the variable assignment in state s; such that $\exists d \in D \ \wedge \ \forall v \in \varphi_\alpha,\ d(v) = s(v)$ where $d(v)$ and $s(v)$ denote the value assignments for variable v in tuple d and state s, respectively.*

Example 2 *Norms, denoted by ι_{escort}, that an NGO is obliged to have an armed escort when entering unsafe regions can be expressed as:*

$$\iota_{escort} = \langle \mathbf{O}, \texttt{area} \in \{16, 21\}, \texttt{escort} \in \{granted\} \rangle.$$

Thus, the example above denotes that regions 16 *and* 21 *should not be entered without an escort (as they are unsafe). Then, the context condition is satisfied when variable* `area` *(indicating the user's location) has the value of* 16 *or* 21.

5.2 Detecting Violations

We say that a state is *relevant* to a norm if the norm's context condition is satisfied in the state. When a state is relevant to a norm, a normative condition is evaluated to determine the state's compliance, which depends on the deontic modality of the norm. Specifically, an obligation is violated if the normative condition μ is not satisfied in state s; *i.e.*, $s \not\models \mu$. Conversely, as prohibitions specify properties that *should not* be realized, a prohibition is violated if the normative condition is satisfied in state s such that $s \models \mu$. Formally,

Definition 3 (Violating State) *Given state s and norm $\iota = \langle \nu, \alpha, \mu \rangle$, a function determining the violation of norm ι in state s is defined as:*

$$violating(s, \iota) = \begin{cases} 1 & \text{if } (s \models \alpha) \wedge (s \not\models \mu) \wedge (\nu = \mathbf{O}) \\ 1 & \text{if } (s \models \alpha) \wedge (s \models \mu) \wedge (\nu = \mathbf{F}) \\ 0 & \text{otherwise}. \end{cases}$$

For instance, considering norm ι_{escort} in Example 2, given state $s = \{(\texttt{area} = 16), (\texttt{escort} = init)\}$ the violation detection function $violation(s, \iota_{escort})$ would return 1, denoting that norm ι_{escort} is violated in state s.

Given a predicted user plan in a plan-tree, the norm reasoner traverses each node in the plan-tree and evaluates the associated user state for any norm violations. Recall from Section 4 that each node in a predicted plan-tree is associated with a user state and an estimated probability of the user visiting the node in the future. Using the estimated probability, the agent selects a set of high-risk norm violations to manage them proactively.

5.3 Finding the Nearest Compliant State

Our assistant agent aims at not only alerting the user of active violations but also proactively steering the user away from those violations that are likely to happen in the future. In order to accomplish this, for each state that violates a norm the agent needs to find a state that is *compliant* with all norms. That is, for each state s where $violating(s, \cdot) = 1$, the agent is to find the nearest state g that satisfies $violating(g, *) = 0$, where $\cdot$ and $*$ are regular expressions denoting any and all, respectively. Here, the distance between two states is measured by the number of variables whose values are different.

Norm violations occur as the result of certain variables in the state space being in particular configurations. Thus, finding compliant states can be intuitively described as a search for alternative value assignments for the variables in the normative condition such that norms are no longer violated. This is analogous to search in constraint satisfaction problems.

When a norm-violating state is detected, the norm reasoner searches the nearby state space by trying out different value assignment combinations for the agent-variables. For each such state, the norm reasoner evaluates the state for norm compliance. The current algorithm is not exhaustive, and only continues the search until a certain number of compliant states are found.

When compliant state g is found for violating state s, state g becomes a new goal state for the agent, generating a planning problem for the agent such that the agent needs to find a series of actions to move from initial state s to goal state g. The goals that fully comply with norms are assigned with *compliance level* 1. When a search for compliant states fails, the agent must proactively decide on remedial actions aimed at either preventing the user from going to a violating state, or mitigating the effects of a violation. In the norm literature these are called *contrary-to-duty obligations* [Prakken and Sergot, 1996]. For instance, a contrary-to-duty obligation in the escort scenario can be defined such that if a user is about to enter a conflict area without an escort, the agent must *alert* the user of the escort requirement. For such partial compliance cases, we assign compliance level 2.

A planning problem can be expressed as a pair of an initial state s and a set of goal states g_i annotated with their compliance levels c_i, such that $\langle s, \{(g_1, c_1)..., (g_m, c_m)\} \rangle$.

Example 3 (Norm Reasoning) *Given a predicted plan-tree in Example 1, if variable* `escort` *for area* 16 *has value $init$ indicating an escort has not been arranged, the agent detects a norm violation and thus searches for a compliant state as follows. Let us define the domain of agent-variable* `escort` *to be: $\{init, requested, granted, denied, alerted\}$. By alternating values, we get the following two compliant states:*

$$\{(granted, 1), (alerted, 2)\},$$

where state $granted$ is fully compliant while state $alerted$ is partially compliant from the agent's perspective, as it complies with the contrary-to-duty obligation to warn the user. As a result, a newly generated planning problem is passed to the planner module as follows:

$$\langle init, \{(granted, 1), (alerted, 2)\} \rangle.$$

6 Planner and Executor

As opposed to precomputing a policy for every possible case, we propose a scalable model where the assistant agent dynamically plans and executes a series of actions to solve smaller problems as they arise. Note that the issues regarding adjustable autonomy are outside the scope of this paper. Instead, we use a cost-based autonomy model where the agent is allowed to execute those actions that do not incur any cost, but is required to get the user's permission to execute costly (or critical) actions.

6.1 Planning

The agent has a set of executable actions, which in the peacekeeping scenario are the following: { `send-request`, `receive-reply`, `alert-user` }. Given a planning problem–*i.e.*, an initial and goal states–from the norm reasoner, the planner module is responsible for finding a series of

actions to accomplish these goals. In Example 3, two goal (or absorbing) states have been assigned by the norm reasoner: an escort is granted or the user is alerted of the need for an escort. Thus, the agent must plan to change the value of escort variable from $init$ to either $granted$ or $alerted$.

Since our representation of planning problems is generic, one may use classical planners in the implementation. Instead, we use an MDP to develop a planner in order to respect uncertainty involved in agent actions, *e.g.*, sending a request may fail due to a communication network failure.

Recall that a predicted user plan from the plan recognizer imposes deadline constraints (specified as the depth of node) to the agent's planning. Specifically, if the user is likely to commit a violation at a certain time step ahead, the agent must take actions to resolve the violation before the time step. In the planner, a deadline constraint is utilized to determine the horizon for an MDP plan solver, such that the agent planner needs to find an optimal policy given the time that the agent has until the predicted violation time.

In Example 3, when the violation is predicted far in advance, an optimal policy prescribes the agent to always request an escort from the other party, except if an escort request has been denied by the other party then the agent should alert the user of the denied request. Note that an optimal policy can change as time elapses, *e.g.*, the user is better off by being warned when there is not enough time left for the agent to arrange an escort. We compare the number of sequential actions in a plan with the depth of node (or the goal's deadline) to determine the plan's feasibility.

The planning problem formulated by the reasoner may not always be solvable; that is, a compliant state can only be accomplished by modifying those variables that the agent does not have access to, or none of the agent's actions has effects that result in the specified goal state. In this case, the agent notifies the user immediately so that the user can take appropriate actions on her own. Otherwise, the agent starts executing its actions according to the optimal policy until it reaches a goal state.

6.2 Execution

Execution of an agent action may change one or more variables. For each newly generated plan (or a policy) from the planner module, an executor is created as a new thread. An executor *waits* on a signal from the *variable observer* that monitors the changes in the environment variables to determine the agent's current state. When a new state is observed the variable observer *notifies* the plan executor to wake up. The plan executor then selects an optimal action in the current state according to the policy and executes the action. After taking an action, the plan executor is resumed to wait on a new signal from the variable observer. If the observed state is an absorbing state, then the plan execution is terminated, otherwise an optimal action is executed from the new state.

The agent's plan can be updated during execution as more recent assessments of rewards arrives from the norm reasoner, forcing the agent to replan. For instance, after the agent requested an escort from the other party, the other party may not reply immediately causing the agent to wait on the request. In the meantime, the user can proceed to make steps towards the unsafe region, imposing a tighter deadline constraint. When the new deadline constraint is propagated to the planner, an optimal policy is updated for the executor, triggering a new action, *e.g.*, to alert the user of the potential violation (instead of trying to arrange an escort).

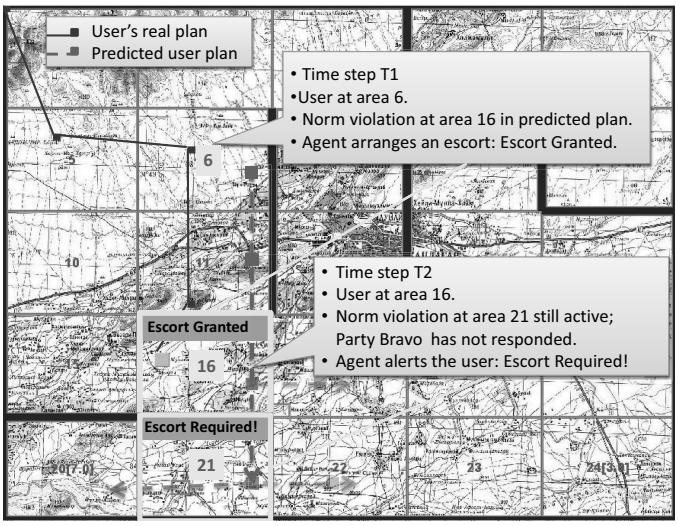

Figure 2: An annotated (and cropped) screenshot of a humanitarian party's planning interface

7 Applications

Through this research, we aim to make not only scientific contributions but also practical impact on real-life applications. The autonomous assistant agent framework that has been presented can be applied to various problem domains. Here, we include some examples of potential applications.

7.1 Military escort planning in peacekeeping

We have implemented our approach as a proof of concept prototype in the context of planning for peacekeeping operations, in a scenario adapted from [Sycara *et al.*, 2010], where two coalition partners (a humanitarian party–Alpha–and a military party–Bravo) plan to operate in the same region according to each party's individual objectives and regulations.

Figure 2 shows the planning interface of a humanitarian party (Alpha), annotated with labels for illustration. At time step $T1$, the agent identifies a norm violation at area 16 in the predicted user plan, for which the agent sends an escort request to Bravo. When the agent receives a reply from Bravo granting a permission the escort status is displayed in the interface. Similarly, the agent sends an escort request for area 21 for another norm violation, but Bravo does not respond. At time step $T2$, an updated policy prescribes the agent to alert the user, and a warning is displayed in the interface.

We have used a simplified military escort planning scenario throughout this paper to illustrate our approach. In practice, the planning and scheduling of escort services in military peacekeeping operations involve complex norm reasoning due to diverse stakeholders. By meeting with the US military and various NGO representatives, we have identified

a significant amount of interest in developing software assistant for this problem domain, and we are currently working on scaling up the system to deal with more realistic settings.

7.2 Potential applications

It is important to note that the goal of this research is not to guide the user in finding optimal planning solutions, but instead, to provide support to the user's planning by identifying and making amends for weaknesses in current plans. As opposed to directing the user to make optimal decisions with respect to a certain objective (as in decision-support systems), we aim to design an agent that can maximize the support to help the user in making decisions based on her own criteria and judgement. Critically, the research presented in this paper is intended to help unburden a user from having to deal with a large number of dynamically changing norms. For instance, in military-civilian collaboration planning, each planner is expected to remember and take into account a large number of directives that change dynamically as the security situation evolves in a war zone. From the user's perspective, independent decision making is crucial, as many rules guiding this kind of collaboration might not necessarily be formalized, so a fully automated planning system would not be suitable.

Furthermore, our research can be applied in many other problem domains such as assistive living technologies for the disabled and the elderly. In this domain, the norms can be defined to specify a set of prohibitions for unsafe activities. When the agent predicts any potential dangers, the agent's new goal becomes restoring a safe state. For instance, if the safe state can be accomplished by taking the agent's available actions, *e.g.*, moving certain objects on the floor, the agent can resolve the issue. When the agent cannot accomplish the goal using its own capabilities, the agent can instead alert the human assistant before an accident happens.

8 Conclusion and Future Work

In this paper, we presented an assistant agent approach to provide prognostic reasoning support for cognitively overloaded human users. We designed the proactive agent architecture by seamlessly integrating several intelligent agent technologies: probabilistic plan recognition, prognostic normative reasoning, and planning and execution techniques. Our approach presents a generic assistant agent framework with which various applications can be built as discussed in Section 7. As a proof of concept application, we implemented a coalition planning assistant agent in a peacekeeping problem domain.

Our approach has several advantages over existing assistant agent approaches. When compared to other decision-theoretic models, our approach is significantly more scalable because of the exponential state space reduction discussed in Section 2. As opposed to assistant agent models where an agent takes turns with the user, our agent has more flexibility in its decision making because the agent can execute multiple plans asynchronously. More importantly, our agent is proactive in that the agent plans ahead of time to satisfy the user's forthcoming needs without a delay. Such proactive assistance is especially an important requirement in time-constrained real-life environments.

We made a specific assumption that agent variables are independent from user variables. We will investigate approaches to relax this assumption. We will also refine the algorithm for determining a plan's feasibility in Section 6.1 by estimating expected time required for each action. Furthermore, we plan to extend our approach to work in a multi-user, multi-agent setting where resolving a norm violation may involve multi-party negotiations. In addition, when there are more than one assistant agents, newly generated goals can be shared or traded among the agents. We will address these special issues raised in multi-agent settings in our future work.

Acknowledgments Research was sponsored by the Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-09-2-0053. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

References

[Boger *et al.*, 2005] J. Boger, P. Poupart, J. Hoey, C. Boutilier, G. Fernie, and A. Mihailidis. A decision-theoretic approach to task assistance for persons with dementia. In *Proc. IJCAI*, pages 1293–1299, 2005.

[Burstein and McDermott, 1996] Mark H. Burstein and Drew V. McDermott. Chapter 17 issues in the development of human-computer mixed-initiative planning. In Barbara Gorayska and Jacob L. Mey, editors, *Cognitive Technology - In Search of a Humane Interface*, volume 113 of *Advances in Psychology*, pages 285 – 303. North-Holland, 1996.

[Fern *et al.*, 2007] A. Fern, S. Natarajan, K. Judah, and P. Tadepalli. A decision-theoretic model of assistance. In *Proc. of AAAI*, 2007.

[García-Camino *et al.*, 2009] A. García-Camino, J.-A. Rodríguez-Aguilar, C. Sierra, and W. W. Vasconcelos. Constraint Rule-Based Programming of Norms for Electronic Institutions. *Journal of Autonomous Agents & Multiagent Systems*, 18(1):186–217, February 2009.

[Modgil *et al.*, 2009] Sanjay Modgil, Noura Faci, Felipe Meneguzzi, Nir Oren, Simon Miles, and Michael Luck. A framework for monitoring agent-based normative systems. In *Proc. of AAMAS*, pages 153–160, 2009.

[Oh *et al.*, 2011] Jean Oh, Felipe Meneguzzi, and Katia Sycara. Probabilistic plan recognition for intelligent information assistants. In *ICAART*, 2011.

[Prakken and Sergot, 1996] Henry Prakken and Marek J. Sergot. Contrary-to-duty obligations. *Studia Logica*, 57(1):91–115, 1996.

[Sycara *et al.*, 2010] K. Sycara, T.J. Norman, J.A. Giampapa, M.J. Kollingbaum, C. Burnett, D. Masato, M. McCallum, and M.H. Strub. Agent support for policy-driven collaborative mission planning. *The Computer Journal*, 53(5):528–540, 2010.