Journal of Artificial Intelligence Research 52 (2015) 361-398 Submitted 7/14; published 3/15

A Bayesian Generative Modeling with Logic-Based Prior

Been Kim BEENKIM@CSAIL.MIT.EDU
Caleb M. Chacha C_CHACHA@CSAIL.MIT.EDU
Julie A. Shah JULIE_A_SHAH@CSAIL.MIT.EDU
Massachusetts Institute of Technology
77 Massachusetts Ave. MA 02139, USA

Abstract

We aim to reduce the burden of programming and deploying autonomous systems to work in concert with people in time-critical domains such as military field operations and disaster response. Deployment plans for these operations are frequently negotiated on-the-fly by teams of human planners. A human operator then translates the agreed-upon plan into machine instructions for the robots. We present an algorithm that reduces this translation burden by inferring the final plan from a processed form of the human team's planning conversation. Our hybrid approach combines probabilistic generative modeling with logical plan validation used to compute a highly structured prior over possible plans, enabling us to overcome the challenge of performing inference over a large solution space with only a small amount of noisy data from the team planning session. We validate the algorithm through human subject experimentations and show that it is able to infer a human team's final plan with 86% accuracy on average. We also describe a robot demonstration in which two people plan and execute a first-response collaborative task with a PR2 robot. To the best of our knowledge, this is the first work to integrate a logical planning technique within a generative model to perform plan inference.

1. Introduction

Robots are increasingly being introduced to work in concert with people in high-intensity domains such as military field operations and disaster response. For example, robot deployment can allow for access to areas that would otherwise be inaccessible to people (Casper & Murphy, 2003; Micire, 2002), to inform situation assessment (Larochelle, Kruijff, Smets, Mioch, & Groenewegen, 2011). The human-robot interface has long been identified as a major bottleneck in utilizing these robotic systems to their full potential (Murphy, 2004). As a result, significant research efforts have been aimed at easing the use of these systems in the field, including careful design and validation of supervisory and control interfaces (Jones, Rock, Burns, & Morris, 2002; Cummings, Brzezinski, & Lee, 2007; Barnes, Chen, Jentsch, & Redden, 2011; Goodrich, Morse, Engh, Cooper, & Adams, 2009). Much of this prior work has focused on ease of use at “execution time.” However, a significant bottleneck also exists in planning the deployment of autonomous systems and in the programming of these systems to coordinate task execution with a human team. Deployment plans are frequently negotiated by human team members on-the-fly and under time pressure (Casper & Murphy, 2002, 2003). For a robot to aid in the execution of such a plan, a human operator must transcribe and translate the result of a team planning session.

In this paper, we present an algorithm that reduces this translation burden by inferring the final plan from a processed form of the human team's planning conversation. Inferring the plan from noisy and incomplete observation can be formulated as a plan recognition problem (Ryall, Marks, & Shieber, 1997; Bauer, Biundo, Dengler, Koehler, & Paul, 2011; Mayfield, 1992; Charniak & Goldman, 1993; Carberry, 1990; Grosz & Sidner, 1990; Gal, Reddy, Shieber, Rubin, & Grosz, 2012). The noisy and incomplete characteristics of observation stem from the fact that not all observed data (e.g., the team's planning conversation) will be a part of the plan that we are trying to infer, and that the entire plan may not be observed. The focus of existing plan recognition algorithms is often to search an existing knowledge base given noisy observation. However, deployment plans for emergency situations are seldom the same, making it infeasible to build a knowledge base. In addition, planning conversations are often conducted under time pressure and are, therefore, often short (i.e., contain a small amount of data). Shorter conversations result in a limited amount of available data for inference, often making the inference problem more challenging.

Our approach combines probabilistic generative modeling with logical plan validation, which is used to compute a highly structured prior over possible plans. This hybrid approach enables us to overcome the challenge of performing inference over a large solution space with only a small amount of noisy data collected from the team planning session.

In this work, we focus on inferring a final plan using text data that can be logged from chat or transcribed speech. Processing human dialogue into more machine-understandable forms is an important research area (Kruijff, Janıcek, & Lison, 2010; Tellex, Kollar, Dickerson, Walter, Banerjee, Teller, & Roy, 2011; Koomen, Punyakanok, Roth, & Yih, 2005; Palmer, Gildea, & Xue, 2010; Pradhan, Ward, Hacioglu, Martin, & Jurafsky, 2004), but we view this as a separate problem and do not focus on it in this paper.

The form of input we use preserves many of the challenging aspects of natural human planning conversations, and can be thought of as noisy observation of the final plan. Because the team is discussing the plan under time pressure, planning sessions often consist of a small number of succinct communications. Our approach can infer the final agreed-upon plan using a single planning session, despite a small amount of noisy data.

We validate the algorithm through experiments with 96 human subjects and show that we are able to infer a human team's final plan with 86% accuracy on average. To the best of our knowledge, this is the first work to integrate a logical planning technique within a generative model to perform plan inference.

In summary, this work includes the following contributions:

- We formulate the novel problem of performing inference to extract a finally agreed-upon plan from a human team planning conversation.

- We propose and validate a hybrid approach to perform this inference that applies the logic-based prior probability over the space of possible agreed-upon plans. This approach performs efficient inference for the probabilistic generative model.

- We demonstrate the benefit of this approach using human team meeting data collected from large-scale human subject experiments (total 96 subjects) and are able to infer a human team's final plan with 86% accuracy on average.

This work extends the preliminary version of this work (Kim, Chacha, & Shah, 2013) to include the inference of complex durative task plans and to infer plans before and after new information becomes available for the human team. In addition, we extend our probabilistic model to be more flexible to different data sets by learning hyper-parameters. We also improve the performance of our algorithm by designing a better proposal distribution for inference.

The formulation of the problem is presented in Section 2, followed by the technical approach and related work in Section 3. Our algorithm is described in Section 4. The evaluation of the algorithm using various data sets is shown in Sections 5 and 6. Finally, we discuss the benefits and limitations of the current approach in Section 7, and conclude with considerations for future work in Section 8.

2. Problem Formulation

Disaster response teams are increasingly utilizing web-based planning tools to plan deployments (Di Ciaccio, Pullen, & Breimyer, 2011). Hundreds of responders access these tools to develop their plans using audio/video conferencing, text chat and annotatable maps. Rather than working with raw, natural language, our algorithm takes a structured form of the human dialogue from these web-based planning tools as input. The goal of this work is to infer a human team's final plan from this human dialogue. In doing so, this work can be used to design an intelligent agent for these planning tools that can actively participate during a planning session to improve the team's decision.

This section describes the formal definition, input and output of the problem. Formally, this problem can be viewed as one of plan recognition, wherein the plan follows the formal representation of the Planning Domain Description Language (PDDL). The PDDL has been widely used in the planning research community and planning competitions (i.e., the International Planning Competition). A plan is *valid* if it achieves a user-specified goal state without violating user-specified plan constraints. Actions may be constrained to execute in *sequence* or in *parallel* with other actions. Other plan constraints can include discrete resource constraints (e.g. the presence of two medical teams) and temporal deadlines for time-durative actions (e.g. a robot can only be deployed for up to 1 hour at a time due to battery life constraints).

We assume that the team reaches agreement on a final plan. The techniques introduced by Kim and Shah (2014) can be used to detect the strength of this agreement, and to encourage the team to further discuss the plan to reach an agreement if necessary. Situations where a team agrees upon a flexible plan with multiple options to be explored will be included in future study. Also, while we assume that the team is more likely to agree on a valid plan, we do not rule out the possibility that the final plan is invalid.

2.1 Algorithm Input

Text data from the human team conversation is collected in the form of utterances, where each utterance is one person's turn in the discussion, as shown in Table 1. The input to our algorithm is a machine-understandable form of human conversation data, as illustrated in the right-hand column of Table 1. This structured form captures the actions discussed and the proposed ordering relations among actions for each utterance.

	Natural dialogue	Structured form (ordered tuple of sets of grounded predicates)
U1	So I suggest using Red robot to cover "upper" rooms (A, B, C, D) and Blue robot to cover "lower" rooms (E, F, G, H).	({ST(rr,A),ST(br,E), ST(rr,B),ST(br,F), ST(rr,C),ST(br,G), ST(rr,D),ST(br,H)})
U2	Okay. so first send Red robot to B and Blue robot to G?	({ST(rr,B),ST(br,G)})
U3	Our order of inspection would be (B, C, D, A) for Red and then (G, F, E, H) for Blue.	({ST(rr,B),ST(br,G)}, {ST(rr,C),ST(br,F)}, {ST(rr,D),ST(br,E)}, {ST(rr,A), ST(br,H)})
U4	Oops I meant (B, D, C, A) for Red. …	({ST(rr,B)},{ST(rr,D)}, {ST(rr,C)},{ST(rr,A)})
U5	So we can have medical crew go to B when robot is inspecting C …	({ST(m,B), ST(r,C)})
U6	First, Red robot inspects B	({ST(r,B)})
U7	Yes, and then Red robot inspects D, Red medical crew to treat B	({ST(r,D),ST(rm,B)})

Table 1: Utterance tagging: Dialogue and structured form examples. (The structured form uses the following shorthand - ST: send to, rr: red robot, br: blue robot, rm: red medical, bm: blue medical, e.g. ST(br,A) : "send the blue robot to room A.")

Although we are not working with raw, natural language, this form of data still captures many of the characteristics that make plan inference based on human conversation challenging. Table 1 shows part of the data using the following shorthand:

ST = SendTo
rr = red robot, br = blue robot
rm = red medical, bm = blue medical
e.g. ST(br, A) = SendTo(blue robot, room A)

2.1.1 Utterance Tagging

An utterance is tagged as an *ordered tuple of sets of grounded predicates*. Following a formal definition for first-order languages, a grounded predicate is an atomic formula whose argument terms are grounded (i.e., no free variables; all variables have an assigned value). In our case, a predicate represents an action applied to a set of objects (a crew member, robot, room, etc.), and an utterance can be represented as ordered sets of these actions. We only consider utterances related to plan formation; greetings and jokes, for example, are not tagged. Each set of grounded predicates represents a collection of actions that, according to the utterance, should happen simultaneously. The order of the sets of grounded predicates indicates the *relative* order in which these collections of actions should happen. For example, ({ST(rr, B), ST(br, G)}, {ST(rm, B)}) corresponds to sending the red robot to room B and the blue robot to room G simultaneously, followed by sending the red medical team to room B.

As indicated in Table 1, the structured dialogue still includes high levels of noise. Each utterance (i.e. U1-U7) discusses a partial plan, and only predicates explicitly mentioned in the utterance are tagged (e.g. U6-U7: the "and then" in U7 implies a sequencing constraint with the predicate discussed in U6, but the structured form of U7 does not include ST(r,B)). Typos and misinformation are tagged without correction (e.g. U3), and any utterances indicating a need to revise information are not placed in context (e.g. U4). Utterances that clearly violate the ordering constraints (e.g. U1: all actions cannot happen at the same time) are also tagged without correction. In addition, information regarding whether an utterance was a suggestion, rejection of or agreement with a partial plan is not coded.

Note that the utterance tagging only contains information about *relative ordering* between the predicates appearing in that utterance, not the *absolute ordering* of their appearance in the final plan. For example, U2 specifies that the two grounded predicates happen at the same time. It does not state when the two predicates happen in the final plan, or whether other predicates will happen in parallel. This simulates how humans conversations often unfold — at each utterance, humans only observe the relative ordering, and infer the absolute order of predicates based on the whole conversation and an understanding of which orderings would make a valid plan. This utterance tagging scheme is also designed to support the future transition to automatic natural language processing. Automatic semantic role labeling (Jurafsky & Martin, 2000), for example, can be used to detect the arguments of predicates from sentences. One of the challenges with incorporating semantic role labeling into our system is that the dialogue from our experiments is often colloquial and key grammatical components of sentences are often omitted. Solving this problem and processing free-form human dialogue into more machine-understandable forms is an important research area, but we view that as a separate problem and do not focus on it in this paper.

2.2 Algorithm Output

The output of the algorithm is an inferred final plan, sampled from the probability distribution over the final plans. The final plan has the same representation to the structured utterance tags (*ordered tuple of sets of grounded predicates*). The predicates in each set represent actions that should happen in parallel, and the ordering of sets indicates the sequence. Unlike the utterance tags, however, the sequence ordering relations in the final plan represent the *absolute* order in which the actions are to be carried out. An example of a plan is $(\{A_1, A_2\}, \{A_3\}, \{A_4, A_5, A_6\})$, where A_i represents a predicate. In this plan, A_1 and A_2 will happen at step 1 of the plan, A_3 happens at step 2 of the plan, and so on.

3. Approach in a Nutshell and Related Work

Planning conversations performed under time pressure exhibit unique characteristics and challenges for inferring the final plan. First, these planning conversations are succinct — participants tend to write shortly and briefly, and to be in a hurry to make a final decision. Second, there may be a number of different valid plans for the team's deployment — even

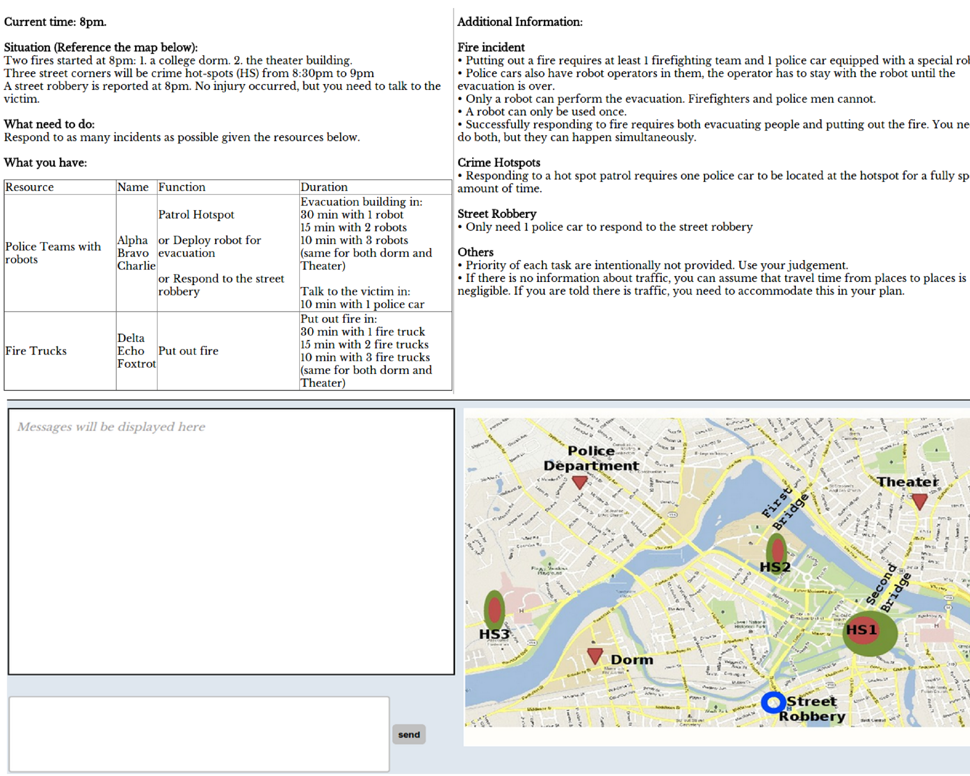


Figure 1: Web-based tool developed and used for data collection

with a simple scenario, people tend to generate a broad range of final plans. This represents the typical challenges faced during real rescue missions, where each incident is unique and participants cannot have a library of plans to choose from at each time. Third, these conversations are noisy — often, many suggestions are made and rejected more quickly than they would be in a more casual setting. In addition, there are not likely to be many repeated confirmations of agreement, which might typically ease detection of the agreed-upon plan.

It might seem natural to take a probabilistic approach to the plan inference problem, as we are working with noisy data. However, the combination of a small amount of noisy data and a large number of possible plans means that inference using typical, uninformative priors over plans may fail to converge to the team's final plan in a timely manner.

This problem could also be approached as a logical constraint problem of partial order planning, if there were no noise in the utterances: If the team were to discuss only partial plans relating to the final plan, without any errors or revisions, then a plan generator or scheduler (Coles, Fox, Halsey, Long, & Smith, 2009) could produce the final plan using global sequencing. Unfortunately, data collected from human conversation is sufficiently noisy to preclude this approach.

These circumstances provided motivation for a combined approach, wherein we built a probabilistic generative model and used a logic-based plan validator (Howey, Long, & Fox, 2004) to compute a highly structured prior distribution over possible plans. Intuitively speaking, this prior encodes our assumption that the final plan is likely, but not required, to be a valid plan. This approach naturally deals with both the noise in the data and the challenge of performing inference over plans with only a limited amount of data. We performed sampling inference in the model using Gibbs and Metropolis-Hastings sampling to approximate the posterior distribution over final plans, and empirical validation with human subject experiments indicated that the algorithm achieves 86% accuracy on average. More details of our model and inference methods are presented in Section 4.

The related work can be categorized into two categories: 1) application and 2) technique. In terms of application, our work relates to plan recognition (Section 3.1). In terms of technique, our approach relates to methods that combine logic and probability, though with different focused applications (Section 3.2).

3.1 Plan Recognition

Plan recognition has been an area of interest within many domains, including interactive software (Ryall et al., 1997; Bauer et al., 2011; Mayfield, 1992), story understanding (Charniak & Goldman, 1993) and natural language dialogue (Carberry, 1990; Grosz & Sidner, 1990).

The literature can be categorized in two ways. The first is in terms of requirements. Some studies (Lochbaum, 1998; Kautz, 1987) require a library of plans, while others (Zhuo, Yang, & Kambhampati, 2012; Ramırez & Geffner, 2009; Pynadath & Wellman, 2000; Sadilek & Kautz, 2010) replace this library with relevant structural information. If a library of plans is required, some studies (Weida & Litman, 1992; Kautz, Pelavin, Tenenberg, & Kaufmann, 1991) assumed that this library can be collected, and that all future plans are guaranteed to be included within the collected library. By contrast, if a library of plans is not required, it can be replaced by related structure information, such as a domain theory or the possible set of actions performable by agents. The second categorization for the literature is in terms of technical approach. Some studies incorporated constraint-based approaches, while others took probabilistic or combination approaches.

First, we reviewed work that treated plan recognition as a knowledge base search problem. This method assumes that you either have or can build a knowledge base, and that your goal is to efficiently search this knowledge base (Lochbaum, 1998; Kautz, 1987). This approach often includes strong assumptions regarding the correctness and completeness of the plan library, in addition to restrictions on noisy data (Weida & Litman, 1992; Kautz et al., 1991), and is applicable in domains where the same plan reoccurs naturally. For example, Gal et al. (2012) studied how to adaptively adjust educational content for a better learning experience, given students' misconceptions, using a computer-based tutoring tool. Similarly, Brown and Burton (1978) investigated users' underlying misconceptions using user data collected from multiple sessions spent trying to achieve the same goal. In terms of technical approach, the above approaches used logical methods to solve the ordering constraints problem of searching the plan library.

We also reviewed work that replaced a knowledge base with the structural information of the planning problem. Zhuo et al. (2012) replaced the knowledge base with action models of the domain, and formulated the problem as one of satisfiability to recognize multi-agent plans. A similar approach was taken by Ramırez and Geffner (2009), wherein action models were used to replace the plan library, while Pynadath and Wellman (2000) incorporated an extension of probabilistic context free grammars (PCFGs) to encode a set of predefined actions to improve efficiency. More recently, Markov logic was applied to model the geometry, motion model and rules for the recognition of multi-agent plans while playing a game of capture the flag (Sadilek & Kautz, 2010). Replacing the knowledge base with structural information reduces the amount of prior information required. However, there are two major issues with the application of prior work to recognize plans from team conversation: First, the above work assumed some repetition of previous plans. For example, learning the weights in Markov logic (which represent the importance or strictness of the constraints) requires prior data from the same mission, with the same conditions and resources. Second, using first-order logic to express plan constraints quickly becomes computationally intractable as the complexity of a plan increases.

In contrast to logical approaches, probabilistic approaches allow for noisy observations. Probabilistic models are used to predict a user's next action, given noisy data (Albrecht, Zuckerman, Nicholson, & Bud, 1997; Horvitz, Breese, Heckerman, Hovel, & Rommelse, 1998). These works use actions that are *normally* performed by users as training data. However, the above approaches do not consider particular actions (e.g., actions that must be performed by users to achieve certain goals within a software system) to be more likely. In other words, while they can deal with noisy data, they do not incorporate structural information that could perhaps guide the plan recognition algorithm. An additional limitation of these methods is that they assume predefined domains. By defining a domain, the set of possible plans is limited, but the possible plans for time-critical missions is generally not a limited set. The situation and available resources for each incident are likely to be different. A method that can recognize a plan from noisy observations, and from an open set of possible plans, is required.

Some probabilistic approaches incorporate structure through the format of a plan library. Pynadath and Wellman (2000) represented plan libraries as probabilistic context free grammars (PCFGs). Then this was used to build Bayes networks that modeled the underlying generative process of plan construction. However, their parsing-based approaches did not deal with partially-ordered plans or temporally interleaved plans. Geib et al. (2008) and Geib and Goldman (2009) overcame this issue by working directly with the plan representation without generating an intermediate representation in the form of a belief network. At each time step, their technique observed the previous action of the agent and generated a pending action set. This approach, too, assumed an existing plan library and relied on the domains with some repetition of previous plans. More recently, Nguyen, Kambhampati, and Do (2013) introduced techniques to address incomplete knowledge of the plan library, but for plan generation rather than plan recognition applications.

Our approach combines a probabilistic approach with logic-based prior to infer team plans without the need for historical data, using only situational information and data from a single planning session. The situational information includes the operators and resources from the domain and problem specifications, which may be updated or modified from one scenario to another. We do not require the development or addition of a plan library to infer the plan, and demonstrate our solution is robust to incomplete knowledge of the planning problem.

3.2 Combining Logic and Probability

The combination of a logical approach with probabilistic modeling has gained interest in recent years. Getoor and Mihalkova (2011) introduced a language for the description of statistical models over typed relational domains, and demonstrated model learning using noisy and uncertain real-world data. Poon and Domingos (2006) proposed statistical sampling to improve searching efficiency for satisfiability testing. In particular, the combination of first-order logic and probability, often referred to as Markov Logic Networks (MLN), was studied. MLN forms the joint distribution of a probabilistic graphical model by weighting formulas in a first-order logic (Richardson & Domingos, 2006; Singla & Domingos, 2007; Poon & Domingos, 2009; Raedt, 2008).

Our approach shares with MLNs the philosophy of combining logical tools with probabilistic modeling. MLNs utilize first-order logic to express relationships among objects. General first-order logic allows for the use of expressive constraints across various applications. However, within the planning domain, enumerating all constraints in first-order logic quickly becomes intractable as the complexity of a plan increases. For example, first-order logic does not allow the explicit expression of action preconditions and postconditions, let alone constraints among actions. PDDL has been well-studied in the planning research community (McDermott, Ghallab, Howe, Knoblock, Ram, Veloso, Weld, & Wilkins, 1998), where the main focus is to develop efficient ways to express and solve planning problems. Our approach exploits this tool to build a highly structured planning domain within the probabilistic generative model framework.

4. Algorithm

This section presents the details of our algorithm. We describe our probabilistic generative model and indicate how this model is combined with the logic-based prior to perform efficient inference. The generative model specifies a joint probability distribution over observed variables (e.g., human team planning conversation) and latent variables (e.g., the final plan); our model learns the distribution of the team's final plan, while incorporating a logic-based prior (plan validation tool). Our key contribution is the design of this generative model with logic-based prior. We also derive the Gibbs sampling (Andrieu, De Freitas, Doucet, & Jordan, 2003) representation and design the scheme for applying Metropolis-Hasting sampling (Metropolis, Rosenbluth, Rosenbluth, Teller, & Teller, 1953) for performing inference on this model.

4.1 Generative Model

We model the human team planning process, represented by their dialogue, as a probabilistic Bayesian model. In particular, we utilize a probabilistic generative modeling approach that has been extensively used in topic modeling (e.g., Blei, Ng, & Jordan, 2003).

We start with a *plan* latent variable that must be inferred through observation of utterances made during the planning session. The model generates each utterance in the conversation by sampling a subset of the predicates in *plan* and computing the relative ordering in which they appear within the utterance. The mapping from the absolute ordering in *plan* to the relative ordering of predicates in an utterance is described in more detail below. Since the conversation is short and the level of noise is high, our model does not distinguish utterances based on the order in which they appear during the conversation. This assumption produces a simple yet powerful model, simplifies the inference steps and enables up to 86% accuracy for the inference of the final plan. However, the model can also be generalized to take the ordering into account with a simple extension. We include further discussion on this assumption and the extension in Section 7.4.

The following is a step-by-step description of the generative model:

1. **Variable** ***plan*****:** The *plan* variable in Figure 2 is defined as an ordered tuple of sets of grounded predicates, and represents the final plan agreed upon by the team. It is distributed over the space of ordered tuples of sets of grounded predicates. We assume that the total number of grounded predicates in one domain is fixed.

   The prior distribution over the *plan* variable is given by:

$$p(plan) \propto \begin{cases} e^{\alpha} & \text{if the plan is valid} \\ 1 & \text{if the plan is invalid.} \end{cases} \tag{1}$$

   where α is a positive number. This models our assumption that the final plan is more likely, but is not necessarily required, to be a valid plan.

   The likelihood of the *plan* is defined as:

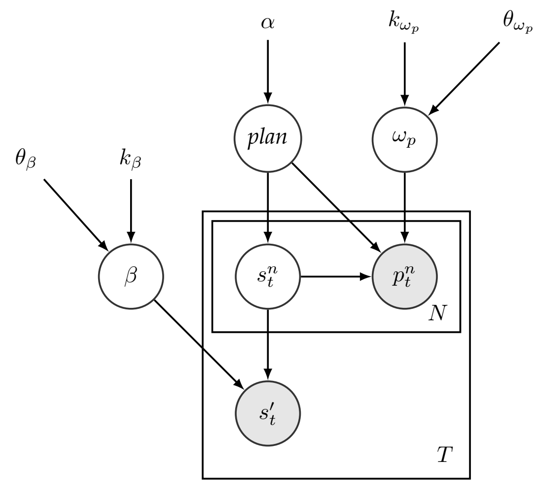


Figure 2: Graphical model representation of the generative model. The *plan* latent variable represents the final agreed-upon plan. The $\boldsymbol{p}_t^n$ variable represents the n^{th} predicate of t^{th} utterance, while $\boldsymbol{s}_t^n$ represents the absolute ordering of that predicate in the *plan*. The $\boldsymbol{s}_t'$ represents the relative ordering of $\boldsymbol{s}^n$ within the utterance t. The latent variable $\boldsymbol{\omega}_p$ represents the noisiness of predicates, and $\boldsymbol{\beta}$ represents the noisiness of the ordering.

$$
\begin{aligned}
p(s, p|plan, \omega_p) &\sim \prod_t \prod_n p(s_t^n, p_t^n|plan, \omega_p) \\
&\sim \prod_t \prod_n p(p_t^n|plan, s_t^n, \omega, p) p(s_t^n|plan)
\end{aligned}
$$

Each set of predicates in *plan* is assigned a consecutive absolute plan step index s, starting at $s = 1$ working from left to right in the ordered tuple. For example, given $plan = (\{A_1, A_2\}, \{A_3\}, \{A_4, A_5, A_6\})$, where each A_i is a predicate, A_1 and A_2 occur in parallel at plan step $s = 1$, and A_6 occurs at plan step $s = 3$.

2. **Variable $\boldsymbol{s}_t^n$:** s_t^n represents a step index (i.e., absolute ordering) of the nth predicate in the tth utterance in the *plan*. A step index represents an *absolute* timestamp of the predicate in the plan. In other words, s_t^n indicates the absolute order of the predicate p_t^n as it appears in *plan*. We use $s_t = \{s_t^1, s_t^2 \dots\}$ to represent a vector of orderings for the tth utterance, where the vector s_t may not be a set of consecutive numbers.

   s_t^n is sampled as follows: For each utterance, n predicates are sampled from *plan*. For example, consider $n = 2$, where the first sampled predicate appears in the second set (i.e., the second timestamp) of *plan* and the second sampled predicate appears in the fourth set. Under these conditions, $s_t^1 = 2$ and $s_t^2 = 4$. The probability of a set being sampled is proportional to the number of predicates contained within that set. For example, given $plan = (\{A_1, A_2\}, \{A_3\}, \{A_4, A_5, A_6\})$, the probability of selecting the first set ($\{A_1, A_2\}$) is $\frac{2}{6}$. This models the notion that people are more likely to discuss plan steps that include many predicates, since plan steps with many actions may require more effort to elaborate. Formally:

$$p(s_t^n = i|plan) = \frac{\#\text{ predicates in set } i \text{ in plan}}{\text{total } \#\text{ of predicates in plan}}. \tag{2}$$

The likelihood of s_t is defined as:

$$\begin{aligned} p(s_t|p_t, s_t^{'}) &\sim p(s_t|plan)p(p_t, s_t^{'}|s_t, \beta, \omega_p, plan) \\ &\sim p(s_t^{'}|s_t, \beta) \prod_n p(s_t^n|plan)p(p_t^n|plan, s_t^n, \omega_p) \end{aligned}$$

3. **Variable s_t' and β:** The variable s_t' is an array of size n, where $s_t' = \{s_t^{'1}, s_t^{'2} \dots s_t^{'n}\}$. The $s_t^{'n}$ random variable represents the relative ordering of the nth predicate within the tth utterance in the *plan*, with respect to other grounded predicates appearing in the tth utterance.

   s_t' is generated from s_t as follows:

$$p(s_t'|s_t) \propto \begin{cases} e^{\beta} & \text{if } s_t' = f(s_t) \\ 1 & \text{if } s_t' \neq f(s_t). \end{cases} \tag{3}$$

   where $\beta > 0$. The β random variable is a hyper-parameter that represents the noisiness of the ordering of grounded predicates appearing throughout the entire conversation. It takes a scalar value, and is sampled from gamma distribution:

$$p(\beta|k_\beta, \theta_\beta) \sim Gamma(k_\beta, \theta_\beta), \tag{4}$$

   where both k_β and θ_β are set to 10.

   The function f is a deterministic mapping from the absolute ordering s_t to the relative ordering s_t'. f takes a vector of absolute plan step indices as input, and produces a vector of consecutive indices. For example, f maps $s_t = (2, 4)$ to $s_t' = (1, 2)$ and $s_t = (5, 7, 2)$ to $s_t' = (2, 3, 1)$.

   This variable models the way predicates and their orders appear during human conversation: People frequently use relative terms, such as "before" and "after," to describe partial sequences of a full plan, and do not often refer to absolute ordering. People also make mistakes or otherwise incorrectly specify an ordering. Our model allows for inconsistent relative orderings with nonzero probability; these types of mistakes are modeled by the value of β.

4. **Variable p_t^n and ω_p:** The variable p_t^n represents the nth predicate appearing in the tth utterance. The absolute ordering of this grounded predicate is s_t^n. The p_t^n is sampled given s_t^n, the *plan* variable and a parameter, ω_p.

The ω_p random variable (hyper-parameter) represents the noisiness of grounded predicates appearing throughout the entire conversation. It takes a scalar value, and is sampled from beta distribution:

$$p(\omega_p|k_{\omega_p}, \theta_{\omega_p}) \sim beta(k_{\omega_p}, \theta_{\omega_p}), \tag{5}$$

where k_{ω_p} is set to 40, and θ_{ω_p} is set to 10.

With probability ω_p, we sample the predicate p_t^n uniformly with replacement from the "correct" set s_t^n in *plan* as follows:

$$p(p_t^n = i|plan, s_t^n = j) = \begin{cases} \frac{1}{\text{\# pred. in set } j} & \text{if } i \text{ is in set } j \\ 0 & \text{o.w.} \end{cases}$$

With probability $1-\omega_p$, we sample the predicate p_t^n uniformly with replacement from "any" set in *plan* (i.e., from all predicates mentioned in the dialogue). Therefore:

$$p(p_t^n = i|plan, s_t^n = j) = \frac{1}{\text{total \# predicates}}. \tag{6}$$

In other words, with higher probability ω_p, we sample a value for p_t^n that is consistent with s_t^n but allows for nonzero probability that p_t^n is sampled from a random plan. This allows the model to incorporate noise during the planning conversation, including mistakes or plans that are later revised.

4.2 Plan Validation Tool

We use the Planning Domain Description Language (PDDL) 2.1 plan validation tool (Howey et al., 2004) to evaluate the prior distribution over possible plans. In this section, we briefly review the PDDL, and the plan validation tool that is used to form the prior in Equation 1.

4.2.1 Planning Domain Description Language

The Planning Domain Description Language (PDDL) (McDermott et al., 1998) is a standard planning language, inspired by the Stanford Research Institute Problem Solver (STRIPS) (Fikes & Nilsson, 1972) and Action Description Language (ADL) (Pednault, 1987), and is now utilized in the International Planning Competition.

A PDDL model of a planning problem has two major components: a domain specification and a problem specification. The domain description consists of a domain-name definition, requirements on language expressivity, definition of object types, definition of constant objects, definition of predicates (i.e. templates for logical facts), and the definition of possible actions that are instantiated during execution. Actions have parameters that may be instantiated with objects, preconditions, and conditional or unconditional effects. An excerpt of the PDDL domain file used in this work, called the RESCUE domain, is shown below. For example, the predicate isSaved encodes the logical fact of whether or not a particular patient has been rescued, and the action SEND−ROBOT is instantiated during execution to send a particular robot to particular location.

```
(define (domain RESCUE)
        (:requirements :typing :durative-actions :negative-preconditions)
        (:types patient valve - thingsToFix location - location
              med-crew mechanic robot - resource)
        (:predicates
              (isAt ?p - thingsToFix ?l - location)
              (isSaved ?p - patient)
              (isFixed ?v - valve)
              (isInspected ?l - location)
              (isAvail ?r - resource)
        )
        (:durative-action SEND-ROBOT
              :parameters (?r - robot ?l - location)
              :duration (= ?duration 1)
              :condition (and
                    (at start (isAvail ?r))
                    (at start (not (isInspected ?l)))
              )
              :effect (and
                    (at start (not (isAvail ?r) ) )
                    (at end (isAvail ?r))
                    (at end (isInspected ?l))
              )
        )
...)
```

The problem specification consists of a problem-name definition, the definition of the related domain-name, the definition of all the possible objects relevant to the planning problem, the initial state of the planning environment as a conjunction of true/false facts, and the definition of goal-states as a logical expression over true/false facts. An excerpt of the PDDL problem specification file used in this work is shown below. The 'init' section describes initial conditions — for example, patient pB is initially situated at location B, and patient pD is at D. The 'goal' section indicates the desired final state — in this case all rooms must be inspected, all patients must be rescued, and all valves fixed. The 'metric' section defines the metric that the planner optimizes when producing a plan.

```
(define (problem rescuepeople)
        (:domain RESCUE)
        (:objects
              pB pD pG − patient
              vC vF − valve
              A B C D E F G H − location
              redMed blueMed − med−crew
              redR blueR − robot
              mech1 − mechanic)
        (:init
              (isAt pB B)
              (isAt pD D)
              ...
              (isAvail redMed)
              (isAvail blueMed)
              (isAvail redR)
              (isAvail blueR)
              (isAvail mech1)
              (not (isSaved pB))
              ...
              (not (isInspected A))
              (not (isInspected B))
              ...
              (not (isFixed vC))
              (not (isFixed vF))
         )
        (:goal (and
              (isSaved pB)
              (isSaved pD)
              ...
              (isFixed vC)
              (isFixed vF)
              (isInspected A)
              (isInspected B)
              ...
              )
         )
        ;(:metric minimize (total−time))
 )
```

For our work, we note that domain specification could be reused from one planning session to another if the capabilities of a team do not change. For example, once the set of possible set of actions is defined in domain specification, it may be sufficient to merely modify the number and names of locations, medical crews, or robots in the problem specification. The domain and the problem specification files represent the only 'prior knowledge' that our approach requires in order to infer the final agreed-upon plan. A valid plan is defined as a totally or partially ordered sequence of grounded predicates that achieves the goal state from the initial state, without violating any constraints. Otherwise, the plan is invalid. The next section describes a plan validation tool that assesses the validity of a given plan, given the domain and problem specification files.

4.2.2 Plan Validation Tool (VAL)

The plan validator is a standard tool that takes as input a planning problem described in PDDL and a proposed solution plan. The tool incorporates three input files: 1) a domain definition file, 2) a problem definition file and 3) a proposed solution plan file. The domain definition file contains types of parameters (e.g., resources, locations), predicate definitions and actions (which also have parameters, conditions and effects). The problem definition file contains information specific to the situation: For example, the number of locations and victims, initial goal conditions and a metric to optimize. A proposed solution plan file contains a single complete plan, described in PDDL. The output of a plan validation tool indicates whether the proposed solution plan is valid (true) or not (false).

Metrics represent ways to compute a plan quality value. For the purpose of this study, the metrics used included: 1) the minimization of total execution time for the radioactive material leakage scenario, and 2) the maximization of the number of incidents responded to in the police incident response scenario. Intuitively speaking, metrics reflect the rational behavior of human experts. It is natural to assume that human experts would try to minimize the total time to completion of time-critical missions (such as in the radioactive material leakage scenario). If first responders cannot accomplish all the necessary tasks in a scenario due to limited availability of resources, they would most likely try to maximize the number of completed tasks (such as in the police incident response scenario).

One could imagine that these input files could be reused in subsequent missions, as the capabilities (actions) of a team may not change dramatically. However, the number of available resources might vary, or there might be rules implicit in a specific situation that are not encoded in these files (e.g., to save endangered humans first before fixing a damaged bridge). In Section 6, we demonstrate the robustness of our approach using both complete and degraded PDDL plan specifications.

The computation of a plan's validity is generally cheaper than that of a valid plan generation. This gives us a way to compute $p(plan)$ (defined in Section 4.1) up to proportionality in a computationally efficient manner. Leveraging this efficiency, we use Metropolis-Hastings sampling, (details described in Section 4.3.1) without calculating the partition function.

4.3 Gibbs Sampling

We use Gibbs sampling to perform inference on the generative model. There are four latent variables to sample: $plan$, the collection of variables s_t^n, ω_p and β. We iterate between sampling each latent variable, given all other variables. The PDDL validator is used when the $plan$ variable is sampled.

4.3.1 Sampling Plan using Metropolis-Hastings

Unlike s_t^n, where we can write down an analytic form to sample from the posterior, it is intractable to directly resample the $plan$ variable, as doing so would require calculating the number of all possible plans, both valid and invalid. Therefore, we use a Metropolis-Hasting (MH) algorithm to sample from the $plan$ posterior distribution within the Gibbs sampling steps.

The posterior of *plan* can be represented as the product of the prior and likelihood, as follows:

$$\begin{aligned} p(plan|s,p) &\propto p(plan)p(s,p|plan) \\ &= p(plan)\prod_{t=1}^{T}\prod_{n=1}^{N} p(s_t^n, p_t^n|plan) \\ &= p(plan)\prod_{t=1}^{T}\prod_{n=1}^{N} p(s_t^n|plan)p(p_t^n|plan, s_t^n) \end{aligned} \tag{7}$$

The MH sampling algorithm is widely used to sample from a distribution when direct sampling is difficult. This algorithm allows us to sample from posterior distribution according to the user-specified proposal distribution without having to calculate the partition function. The typical MH algorithm defines a proposal distribution, $Q(x'|x_t)$, which samples a new point (i.e., x': a value of the *plan* variable in our case) given the current point x_t. The new point can be achieved by randomly selecting one of several possible moves, as defined below. The proposed point is then accepted or rejected, with a probability of $\min(1, \text{acceptance ratio})$.

Unlike simple cases, where a Gaussian distribution can be used as a proposal distribution, our distribution needs to be defined over the plan space. Recall that *plan* is represented as an ordered tuple of sets of predicates. In this work, the new point (i.e., a candidate plan) is generated by performing one of the following moves on the current plan:

- Move to next: Randomly select a predicate that is in the current plan, and move it to the next timestamp. If it is in the last timestamp in the plan, move it to the first timestamp.
- Move to previous: Randomly select a predicate that is in the current plan, and move it to the previous timestamp. If it is in the first timestamp in the plan, move it to the last timestamp.
- Add a predicate to plan: Randomly select a predicate that is not in the current plan, and randomly choose one timestamp in the current plan. Add the predicate to the chosen timestamp.
- Remove a predicate from plan: Randomly select and remove a predicate that is in the current plan.

These moves are sufficient to allow for movement from one arbitrary plan to another. The intuition behind designing this proposal distribution is described in Section 7.5.

Note that the proposal distribution, as it is, is not symmetrical — $Q(x'|x_t) \neq Q(x_t|x')$. We need to compensate for that according to the following,

$$p^*(x')Q(x'|x_t) = p^*(x)Q(x_t|x'), \tag{8}$$

where p^* is the target distribution. This can be done simply by counting the number of moves possible from x' to get to x, and from x' to x, and weighing the acceptance ratio such that Equation 8 is true. This is often referred to as Hastings correction, which is performed to ensure that the proposal distribution does not favor some states over others.

Next, the ratios of the proposal distribution at the current and proposed points are calculated. When $plan$ is valid, $p(plan)$ is proportional to e^{α}, and when $plan$ is invalid, it is proportional to 1, as described in Equation 1. Plan validity is calculated using the plan validation tool. The remaining term, $p(s_t^n|plan)p(p_t^n|plan, s_t^n)$, is calculated using Equations 2 and 6.

Then, the proposed $plan$ is accepted with the following probability: $\min\left(1, \frac{p^*(plan=x'|s,p)}{p^*(plan=x_t|s,p)}\right)$, where p^* is a function proportional to the posterior distribution.

Although we chose to incorporate MH, it is not the only usable sampling method. Any other method that does not require calculation of the normalization constant (e.g., rejection sampling or slice sampling) could also be used. However, for some methods, sampling from the ordered tuple of sets of grounded predicates can be slow and complicated, as pointed out by Neal (2003).

4.3.2 Sampling Hyper-Parameters β and ω_p with Slice Sampling

We use slice sampling to sample both β and ω_p. This method is simple to implement and works well with scalar variables. Distribution choices are made based on the valid value each can take. β can take any value, preferably with one mode, while ω_p can only take a value between $[0, 1]$. MH sampling may also work; however, this method could be overly complicated for a simple scalar value. We chose the stepping out procedure, as described by Neal et al (Neal, 2003).

4.3.3 Sampling s_t^n

Fortunately, an analytic expression exists for the posterior of s_t^n:

$$\begin{aligned} p(s_t|plan, p_t, s_t') &\propto p(s_t|plan)p(p_t, s_t'|plan, s_t) \\ &= p(s_t|plan)p(p_t|plan, s_t)p(s_t'|s_t) \\ &= p(s_t'|s_t)\prod_{n=1}^{N} p(s_t^n|plan)p(p_t^n|plan, s_t^n) \end{aligned}$$

Note that this analytic expression can be expensive to evaluate if the number of possible values of s_t^n is large. In that case, one can marginalize s_t^n, as the variable we truly care about is the $plan$ variable.

5. Experimentation

In this section, we explain the web-based collaboration tool that is used in our experiment and two fictional rescue scenarios given to human subjects in the experiment.

5.1 Web-Based Tool Design

Disaster response teams are increasingly using web-based tools to coordinate missions and share situational awareness. One of the tools currently used by first responders is the Next

Generation Incident Command System (NICS) (Di Ciaccio et al., 2011). This integrated sensing and command-and-control system enables the distribution of large-scale coordination across multiple jurisdictions and agencies. It provides video and audio conferencing capabilities, drawing tools and a chat window, and allows for the sharing of maps and resource information. Overall, the NICS enables the collection and exchange of information critical to mission planning.

We designed a web-based collaboration tool modeled after this system, with a modification that requires the team to communicate solely via text chat. This tool was developed using Django (Holovaty & Kaplan-Moss, 2009), a free and open-source Web application framework written in Python. Django is designed to ease working with heavy-duty data, and provides a Python API to enable rapid prototyping and testing. Incoming data can be easily maintained through a user-friendly administrative interface. Although it is a simplified version of the NICS, it provides the essence of the emerging technology for large-scale disaster coordination (Figure 1).

5.2 Scenarios

Human subjects were given one of two fictional rescue scenarios and asked to formulate a plan by collaborating with their partners. We collected human team planning data from the resulting conversations, and used this data to validate our algorithm. The first scenario involves a radioactive material leakage accident in a building with multiple rooms, where all tasks (described below) were assumed to take one unit of time. We added complexity to the scenario by announcing a new piece of information halfway through the planning conversation, requiring the team to change their plan. The second scenario also included time-durative actions (e.g., action A can only take place if action B is taking place). These scenarios are inspired by those described in emergency response team training manuals (FEMA, 2014), and are designed to be completed in the reasonable time for our experiments.

5.2.1 First Scenario: Radioactive Material Leakage

This disaster scenario involves the leakage of radioactive material on a floor consisting of eight rooms. Each room contains either a patient requiring in-person assessment or a valve that must be repaired (Figure 4).

Goal State: All patients are assessed in-person by a medical crew. All valves are fixed by a mechanic. All rooms are inspected by a robot.

Constraints: There are two medical crews, red and blue (discrete resource constraint), one human mechanic (discrete resource constraint) and two robots, red and blue (discrete resource constraint). For safety purposes, a robot must inspect the radioactivity of a room before human crews can be sent inside (sequence constraint).

Assumption: All tasks (e.g. inspecting a room, fixing a valve) take the same amount of time (one unit), and there are no hard temporal constraints. This assumption was made to conduct the initial proof-of-concept experimentation described in this paper, and is relaxed in the scenario described in Section 5.2.2.

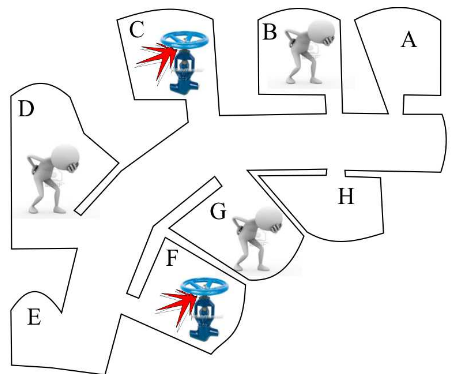


Figure 3: Radioactive material leakage scenario

Announcement: During the planning session, the team receives a situation update that the red robot is now out of order, requiring the team to modify their previously discussed plan to only use one robot for deployment. The announcement triggers automatically once the team has exchanged 20 utterances. The purpose of this announcement is to increase task complexity for the team, to have at least two competing plans and to increase the level of noise in the conversation.

This scenario produces a large number of possible plans (more than 10^{12}), many of which are valid for achieving the goals without violating the constraints.

5.2.2 Second Scenario: Police Incidents Response

The second scenario involves a team of police officers and firefighters responding to a series of incidents occurring in different time frames. This scenario includes more complicated time-durative actions than the first, as well as interdependency of tasks that has to be taken into account when planning. The current time is given as 8 p.m. Two fires have started at this time: one at a college dorm and another at a theater building, as shown in Figure 4. Also, three street corners, indicated as crime hot-spots (places predicted to experience serious crimes, based on prior data), become active between 8:30 p.m. and 9 p.m. There is also a report of a street robbery taking place at 8 p.m. No injury has occurred; however, a police officer must speak with the victim to file an incident report.

Goal State: Respond to as many incidents as possible given the resources listed in Table 2.

Constraints:

- Putting out a fire requires one fire truck and one police car equipped with a robot.
- A police car must stay with the robot until an evacuation is over.
- Only a robot can perform an evacuation.
- Each robot can only be used once.
- Successfully responding to a fire requires both evacuating the building and putting out the fire. Both actions can happen simultaneously.

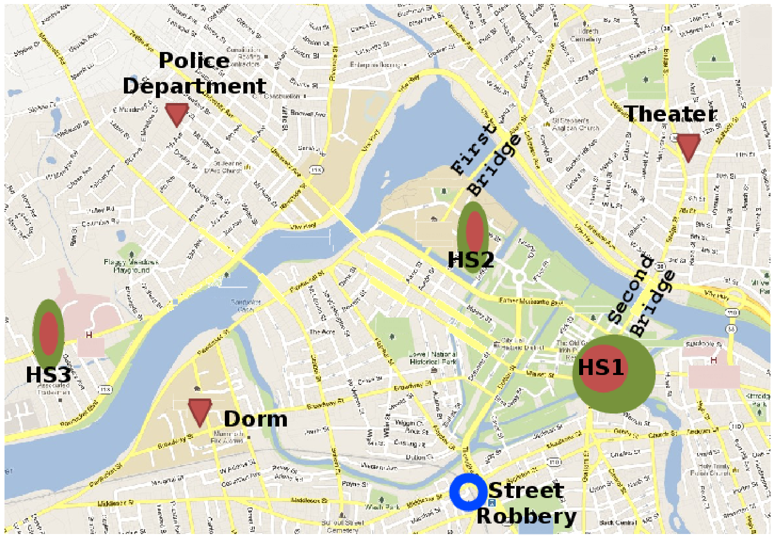


Figure 4: Police incident response scenario

Resources	Name	Function	Duration
Police teams with robots	Alpha Bravo Charlie	Patrol hotspot Deploy robot for evacuation Respond to the street robbery	Evacuate one building in: 30 min with one robot 15 min with two robots 10 min with three robots Talk to the victim in: 10 min with one police car
Fire trucks	Delta Echo Foxtrot	Put out fire	Put out fire in: 30 min with one fire truck 15 min with two fire trucks 10 min with three fire trucks (same for both dorm and theater)

Table 2: Resources available in police incident response scenario

- Responding to a hot-spot patrol requires one police car to be located at the site for a specified amount of time.
- Only one police car is necessary to respond to the street robbery.

Assumption and Announcement: If no information about traffic is provided, the travel time from place to place is assumed to be negligible. During the planning session, the team receives the following announcement: "The traffic officer just contacted us, and said the First and Second bridges will experience heavy traffic at 8:15 pm. It will take at least 20 minutes for any car to get across a bridge. The travel time from the theater to any hot-spot is about 20 minutes without using the bridges." Once this announcement is made, the team must account for the traffic in their plan.

6. Evaluation

In this section, we evaluate the performance of our plan inference algorithm through initial proof-of-concept human subject experimentation, and show we are able to infer a human team's final plan with 86% accuracy on average, where "accuracy" is defined as a composite measure of task allocation and plan sequence accuracy measures. We also describe a robot demonstration in which two people plan and execute a first-response collaborative task with a PR2 robot.

6.1 Human Team Planning Data

As indicated previously, we designed a web-based collaboration tool modeled after the NICS system (Di Ciaccio et al., 2011) used by first-response teams, but with a modification that requires the team to communicate solely via text chat. For the radioactive material leakage scenario, before announcement, 13 teams of two (a total of 26 participants) were recruited through Amazon Mechanical Turk and from the greater Boston area. Recruitment was restricted to those located in the US to increase the probability that participants were fluent in English. For the radioactive material leakage scenario, after announcement, 21 teams of two (a total of 42 participants) were recruited through Amazon Mechanical Turk and from the greater Boston area. For the police incident response scenario, 14 teams of two (total 28 participants) were recruited from the greater Boston area. Participants were not required to have prior experience or expertise in emergency or disaster planning, and we note that there may be structural differences in the dialog of expert and novice planners. We leave this topic for future investigation.

Each team received one of the two fictional rescue scenarios described in Section 5.2, and was asked to collaboratively plan a rescue mission. Upon completion of the planning session, each participant was asked to summarize the final agreed-upon plan in the structured form described previously. An independent analyst reviewed the planning sessions to resolve discrepancies between the two members' descriptions when necessary. The first and second authors, as well as two independent analysts, performed utterance tagging, with each team planning session tagged and reviewed by two of these four analysts. On average, 36% of predicates mentioned per data set did not end up in the final plan.

6.2 Algorithm Implementation

The algorithm was implemented in Python, and the VAL PDDL 2.1 plan validator (Howey et al., 2004) was used. We performed 2,000 Gibbs sampling steps on the data from each planning session. The initial *plan* value was set to two to five moves (from MH proposal distribution) away from the true plan. The initial value for s variable was randomly set to any timestamp in the initial *plan* value.

Within one Gibbs sampling step, we performed 30 steps of the Metropolis-Hastings (MH) algorithm to sample the plan. Every 20 samples were selected to measure accuracy (median), after a burn-in period of 200 samples.

Results We assessed the quality of the final plan produced by our algorithm in terms of the accuracy of task allocation among agents (e.g. which medic travels to which room) and the accuracy of the plan sequence.

Two metrics for task allocation accuracy were evaluated: 1) The percent of inferred plan predicates appearing in the team's final plan [% Inferred], and 2) the percent noise rejection of extraneous predicates that were discussed but do not appear in the team's final plan [% Noise Rej].

We evaluated the accuracy of the plan sequence as follows: A pair of predicates is *correctly ordered* if it is consistent with the relative ordering in the true final plan. We measured the percent accuracy of sequencing [% Seq] by $\frac{\text{\# correctly ordered pairs of correct predicates}}{\text{total \# of pairs of correct predicates}}$. Only correctly estimated predicates were compared, as there is no ground truth relation for predicates not included in the true final plan. We used this relative sequencing measure because it does not compound sequence errors, as an absolute difference measure would (e.g. where an error in the ordering of one predicate early in the plan shifts the position of all subsequent predicates).

Overall "composite" plan accuracy was computed as the arithmetic mean of the task allocation and plan sequence accuracy measures. This metric summarizes the two relevant accuracy measures so as to provide a single metric for comparison between conditions. We evaluated our algorithm under four conditions: 1) perfect PDDL files, 2) PDDL problem file with missing goals/constants (e.g. delete available agents), 3) PDDL domain file missing a constraint (e.g. delete precondition), and 4) using an uninformative prior over possible plans.

The purpose of the second condition, PDDL problem file with missing goals/constants, was to test the robustness of our approach to incomplete problem information. This PDDL problem specifiction was intentionally designed to omit information regarding one patient (pG) and one robot (blueR). It also omitted the following facts about the initial state: that pG was located at G, the blueR was available to perform inspections, and patient pG patient was not yet rescued. The goal state omitted that pG patient was to be rescued. This condition represented a significant degradation of the problem definition file, since the original planning problem involved only three patients and two robots.

The purpose of the third condition, PDDL domain file with a missing constant, was to test the robustness of our approach to missing constraints (or rules for successful execution). It is potentially easy for a person to miss specifying a rule that is often implicitly assumed. The third condition omitted the following constraint from the domain file: all the rooms are to be inspected prior to sending any medical crews. This condition represented a significant degradation of the domain file, since this constraint affected any action involving one of the medical crew teams.

Results shown in Tables 3-5 are produced by sampling *plan* and s variables and fixing $\beta = 5$ and $\omega_p = 0.8$. The tables report median values for the percent of the inferred plan predicates appearing in the final plan [% Inferred], noise rejection [% Noise Rej.], and sequence accuracy [% Seq.]. We show that our algorithm infers final plans with greater than 86% composite accuracy on average. We also show that our approach is relatively robust to degraded PDDL specifications (i.e., PDDL with missing goals, constants and constraints). Further discussion of sampling hyper-parameters is found in Section 7.2.

6.3 Concept-of-Operations Robot Demonstration

We illustrate the use of our plan inference algorithm through a robot demonstration in which two people plan and execute a first-response collaborative task with a PR2 robot. The participants plan an impending deployment using the web-based collaborative tool we developed. Once the planning session is complete, the dialogue is tagged manually. The plan inferred from this data is confirmed with the human planners and provided to the robot for execution. The registration of predicates to robot actions, and room names to map locations, is performed offline in advance. While the first responders are on their way to the accident scene, the PR2 autonomously navigates to each room, performing online localization, path planning and obstacle avoidance. The robot informs the rest of the team as it inspects each room and confirms it is safe for human team members to enter. Video of this demo can be found here: `http://tiny.cc/uxhcrw`.

7. Discussion

In this section we discuss the results and trends in Tables 3-5. We then discuss how sampling hyper-parameters improves inference accuracy, and provide an interpretation of inferred hyper-parameter values and how they relate to data characteristics. We also provide additional support for the use of PDDL by analyzing multiple Gibbs sampling runs. Our rationale behind the i.i.d assumption on utterances made in the generative model is explained, and we show how a simple extension to our model can relax this assumption. Finally, we provide our rationale for designing the proposal distribution for the sampling algorithm.

7.1 Results

The average accuracy of the inferred final plan improved across all three scenarios with the use of perfect PDDL as compared to an uninformative prior over possible plans. The sequence accuracy also consistency improved with the use PDDL, regardless of noise level or the type of PDDL degradation. The three scenarios exhibited different levels of "noise," defined as the percentage of utterances that did not end up in the finally agreed upon plan. The police incidents response scenario produced substantially higher noise (53%), as compared to the radioactive material leaking scenario before announcement (38%) and after announcement (17%). This is possibly because the police incidents scenario included durative-actions, whereas the others did not. Interestingly, perfect PDDL produced more

	Task Allocation		% Seq.	Composite % Acc.
	% Inferred	% Noise Rej.		
PDDL	61	100	97	86
PDDL with missing goals and constants	100	58	77	78
PDDL with missing constraint	70	100	87	86
No PDDL	70	58	66	65

Table 3: Radioactive material leakage scenario plan accuracy results, before announcement (13 teams / 26 subjects). The table reports median values for the percent of the inferred plan predicates appearing in the final plan [% Inferred], noise rejection [% Noise Rej.], and sequence accuracy [% Seq.]. Composite % Accuracy is calculated as the average of the previous three measures.

	Task Allocation		% Seq.	Composite % Acc.
	% Inferred	% Noise Rej.		
PDDL	77	100	83	87
PDDL with missing goals and constants	100	54	97	84
PDDL with missing constraint	72	100	90	87
No PDDL	100	54	81	78

Table 4: Radioactive material leakage scenario plan accuracy results, after announcement (21 teams / 42 subjects). The table reports median values for the percent of the inferred plan predicates appearing in the final plan [% Inferred], noise rejection [% Noise Rej.], and sequence accuracy [% Seq.]. Composite % Accuracy is calculated as the average of the previous three measures.

	Task Allocation		% Seq.	Composite % Acc.
	% Inferred	% Noise Rej.		
PDDL	97	89	97	86
PDDL with missing goals and constants	92	86	92	83
PDDL with missing constraint	97	89	97	85
No PDDL	81	95	81	82

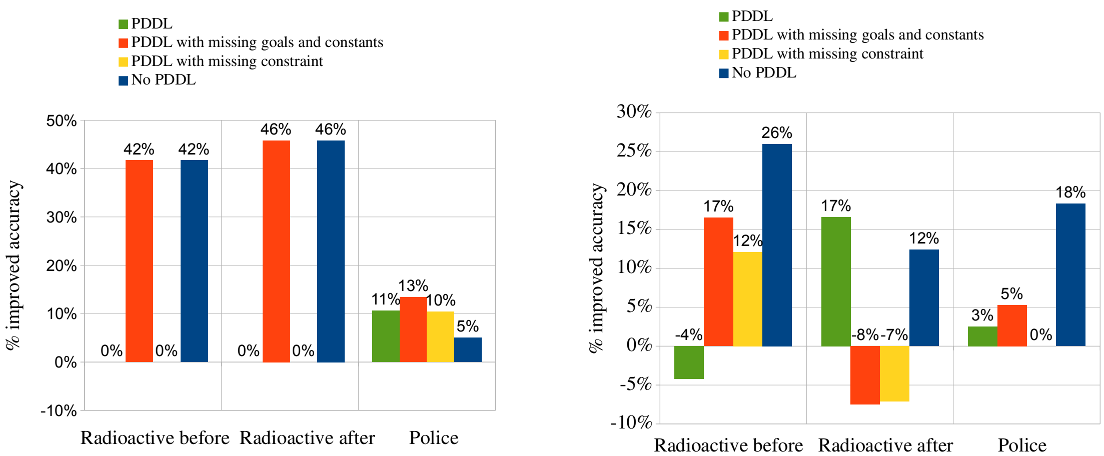
Table 5: Police incidents response scenario plan accuracy results (14 teams / 28 subjects). The table reports median values for the percent of the inferred plan predicates appearing in the final plan [% Inferred], noise rejection [% Noise Rej.], and sequence accuracy [% Seq.]. Composite % Accuracy is calculated as the average of the previous three measures.

substantial improvements in sequence accuracy when noise level was higher, in the radioactive material leaking scenario before announcement, and in police incidents scenario.

Accuracy in task allocation, on the other hand, did differ depending on the noise level and the type of PDDL degradation. The noise rejection ratio was the same or better with PDDL or PDDL with a missing constraint, as compared to an uninformative prior, for scenarios with less noise (e.g. the radioactive material leaking scenarios before and after announcement). However, PDDL did not provide benefit to the noise rejection ratio for the police incidents scenario where the noise level was more than 50%. However, in this case PDDL did provide improvements in inferred task allocation.

7.2 Sampling Hyper-Parameters

This section discusses the results of hyper-parameter sampling. First, we show that each data point (i.e., each team's conversation) converges to different hyper-parameter values, then show that those values capture the characteristics of each data point. Second, we show how learning different sets of hyper-parameters improves different measures of accuracy, and describe how this is consistent with our interpretation of the hyper-parameters in our model.



(a) Improvements in noise rejection when sampling ω_p

(b) Improvements in sequence accuracy when sampling β

Figure 5: Percent improvements in median noise rejection and median sequence accuracy when sampling hyper-parameters versus setting $\omega_p = 0.8$ and $\beta = 5$.

7.2.1 Improvements in Sequence Accuracy versus Noise Rejection

The hyper-parameter β represents the noisiness in predicate ordering, while the hyper-parameter ω_p represents the noisiness in the assignment of predicates. Setting these parameters to a fixed value corresponds to an assumption about the noisiness of the data set. We can learn these parameters through Gibbs sampling, allowing these values to be adjusted according to different characteristics of each data set. The details of how we sample hyper-parameters are explained in Section 4.3.2. We performed 2,000 Gibbs sampling steps on the data from each planning session. The initial values of ω_p and β were sampled from their prior, where the parameters were set to the values described in Section 4.1.

We found that when we learned ω_p (with $\beta = 5$), the noise rejection rate improved compared with when we fixed $\omega_p = 0.8$. In the radioactive material leakage scenario, both before and after the mid-scenario announcement, the noise rejection ratio was improved by as much as 41% and 45%, respectively; in the police incident response scenario, we observed up to a 13% improvement (Figure 5a). Note that in all cases the median noise rejection ratio was maintained or improved with the sampling of ω_p.

Similarly, when we learned β (with $\omega_p = 0.8$), sequence accuracies generally improved. In the radioactive material leakage scenario, before and after the announcement, the sequence accuracy improved by up to 26% and 16%, respectively; in the police incident response scenario, we observed up to an 18% improvement (Figure 5b). Note that three cases did see a degradation in accuracy of up to 4-8%. However in nine out of the twelve cases the sequence accuracy was maintained or improved with the sampling of β.

Interestingly, most of the samples achieved the highest overall composite accuracy when only *plan* and s were learned, and the hyper-parameters were fixed. In particular, we observed an average 5% (± 3%) decrease in composite accuracy when sampling all four variables together. One of the possible explanations for this finding is that, due to the limited amount of data, Gibbs sampling may require many more iterations to converge all the variables. This result suggests that one may choose the set of hyper-parameters to learn based on which measure of accuracy is more important to the user.

7.2.2 Interpretation of Inferred Values of Hyper-Parameters

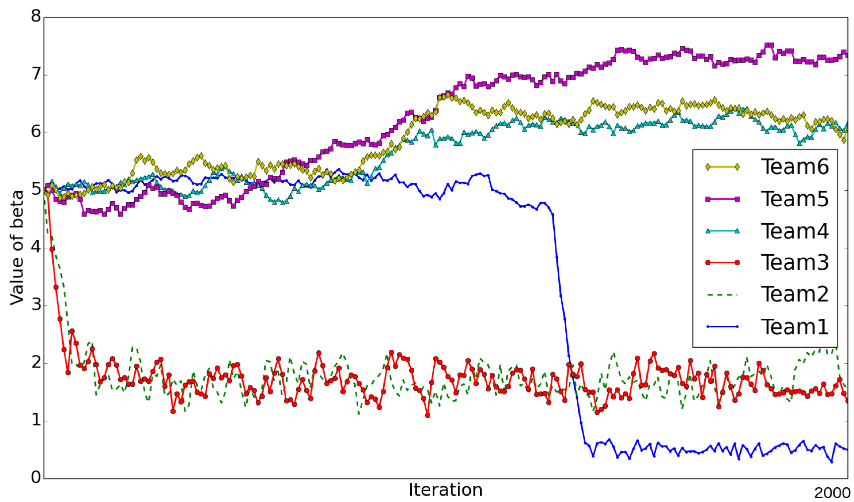
As described in Section 4.1, the ω_p parameter models the level of noise in predicates within the data. In other words, the ω_p parameter is designed to model how many suggestions the team makes during the conversation that are subsequently included in the final plan. If the noise level is high, a lower-valued ω_p will represent the characteristics of the conversation well, which may allow for better performance. (However, if the noise level is too high, the inference may still fail.)

To compare the learned value of ω_p with the characteristics of the conversation, we need a way to calculate how noisy the conversation is. The following is one way to manually estimate the value of ω_p: First, count the utterances that contain any predicates. Then, count the utterances that contain predicates included in the final plan. The ratio between these two numbers can be interpreted as noisiness in the predicates; the lower the number, the more the team talked about many possible plans.

We performed this manual calculation for two teams' trials — Team 3 and 10 — to compare their values to the learned values. In Team 3's trial, only 19.4% of the suggestions made during the conversation were included in the final plan (i.e., almost 80% of suggestions were not relevant to the final plan). On the other hand, 68% of suggestions made in Team 10's trial were included in the final plan. Using this interpretation, Team 3's trial is more than twice as noisy as Team 10's trial.

The converged value of ω_p is lower in Team 3's trial than in Team 10's trial, reflecting the characteristics of each data set. Figure 6b shows the converged value of ω_p for each team's trial (sub-sampled, and for a subset of the dataset). The figure presents values of

(a) Examples of β value convergences



(b) Examples of ω_p value convergence

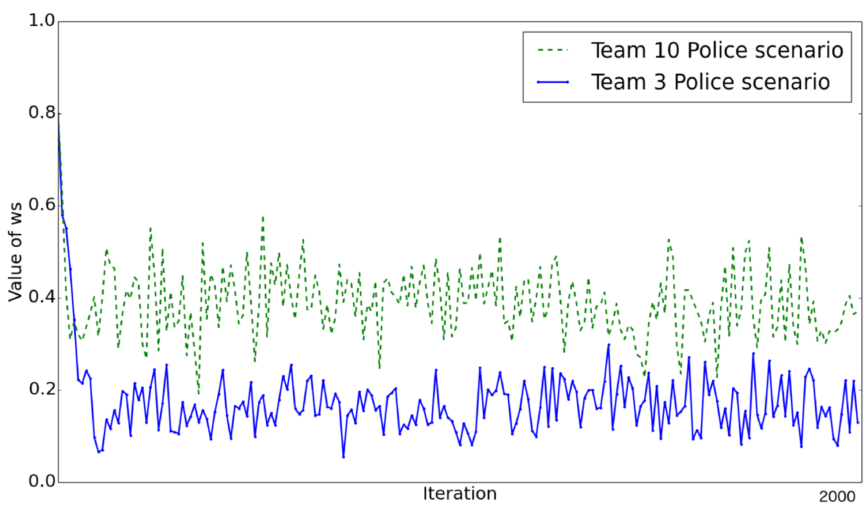


Figure 6: Inferred values of hyper-parameters (only showing subset of data set)

ω_p at each iteration of the Gibbs sampling step. Note that the samples from Team 3's trial converge to 20%, while the samples from Team 10's trial converge to 40%. The lower value of ω_p represents higher noise level, and matches our intuition.

However, there is no conclusive way to prove that these converged values are the true values. In theory, the Gibbs sampling algorithm only guarantees convergences to the true value with an infinite number of iterations. Therefore, we cannot prove that the converged ω_p variables shown in Figure 6 are the true values. In practice, a trace plot, such as that in Figure 6, is drawn in order to demonstrate convergence to a local optimum. The fact that the values appear to plateau after a burn-in period provides support of convergence to a local optimum point. Investigation of this potentially local optimum point suggests that the ω_p value for each data point can be different, and that we can observe some relationship between the ω_p value and the characteristics of the data set. In addition, the manual calculation of 'noisiness' is only one way of interpreting the 'noisiness' of the data set. Therefore, this analysis should be considered as one possible way to gain insight into the learned values; not a rigorous proof of the relation between the learned value of the hyper-parameter and the characteristics of the data.

7.3 The Benefit of PDDL

This section provides additional evidence of the benefit of using PDDL by analyzing multiple runs using the same data and sampling algorithm. As explained in Section 4.3, Gibbs sampling is an approximate inference algorithm that can produce different results on each run.

In this section we evaluate runs over a wide range of different settings to show that the benefit of PDDL applies not just to a particular setting of parameters, but also to different settings. We analyzed three cases across a range of parameters: 1) learning both *plan* and s, 2) learning *plan*, s and ω_p and 3) learning *plan*, s and β. In the first case, we changed the value of α to range from 3 to 1,000, ω_p from 0.3 to 0.8, and β from 1 to 100. In the second case, in addition to α and β parameters, we varied the parameters for the prior distribution of ω_p — k_{ω_p} and θ_{ω_p}; both ranging from 2 to 70. In the third case, in addition to α and ω_p parameters, we varied the parameters for the prior distribution of β — k_β and θ_β; both ranging from 0.1 to 50. Values from the all ranges were selected randomly to produce a total of 613 runs.

Eighty-two percent of the 613 runs showed higher accuracy when PDDL was used than when PDDL was not used. This suggests that adding the structured prior improves accuracy over a wide range of parameter settings. Figure 7 presents the ratio of runs that saw benefit from the use of the PDDL, for each of the three scenarios.

Interestingly, the highest accuracy was not always achieved with perfect PDDL files; in some cases, the highest accuracy was achieved with imperfect PDDL files (e.g., PDDL file with missing goals/constraints, as described in Section 6). This observation may be explained by the possibility that some finally agreed-upon plans 1) are not complete and/or 2) violate constraints (mostly due to participants' misunderstandings). For example: Prior to the announcement during the radioactive material leakage scenario, a number of teams had not finished building complete plans. Therefore, the final plans in these cases may have been better inferred with incomplete PDDL files (consistent with Table 4). In the police incident response scenario, however, a number of teams missed the constraint that the hot-spot patrolling task is only considered complete if that hot-spot is fully covered from 8:30 p.m. to 9 p.m. A number of teams dispatched police cars only for a portion of that time window, resulting in invalid plans with the perfect PDDL files (consistent with Table 5)

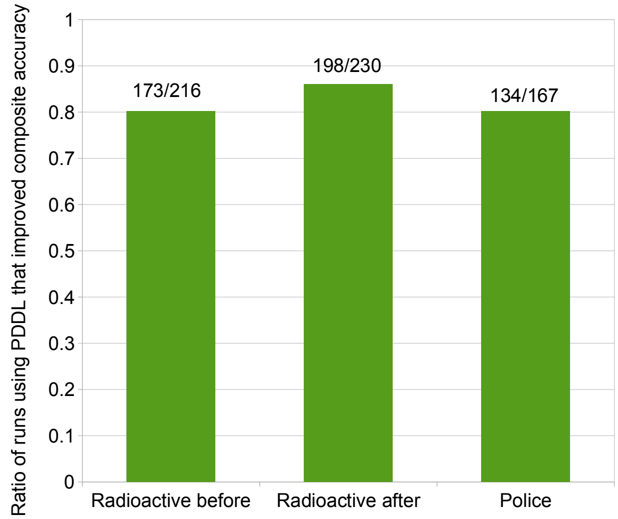


Figure 7: Ratio of runs that show the benefit of using PDDL

The improvements achieved by adding the structure to the prior using PDDL suggest that the structural information is beneficial to our inference problem. It would be interesting to systematically investigate the smallest set of structural information that achieves accuracy improvements, given a fixed computation budget, in future work.

7.4 The i.i.d Assumption on the Utterance in the Generative Model

Our generative model considers that all utterances are independent and identically distributed samples from the plan variable. In other words, we consider that all utterances give equal evidence to a plan, regardless of the order in which they appear during the conversation. An alternative would be to have a different weight for each utterance, to take the ordering into account. In this section, we explain the reasons for the i.i.d. assumption, and how a simple extension to the current model can relax this assumption.

In the human subject data collected for this work, we did not observe a clear relationship between the order of an utterance and whether the suggestion is included in the final plan. For example, a number of teams decided to include parts of a plan that were discussed at the beginning of the conversation within their final plan, after discussing many other possibilities. The distribution of the utterances included in the final plan is shown in Figure 8. In addition, when a team discusses plans under time pressure, the planning sessions often consist of a small number of succinct communications. For example, the average number of predicates in all utterances in a planning session is 90, whereas the average number of predicates in a final plan is 12. A succinct conversation yields less available data for the inference; therefore, a complicated model may fail to correctly infer all the latent variables. A time series model, wherein the ordering is taken into account and the weight of each utterance is a latent variable that needs to be learned from the data, is an example of such a model.

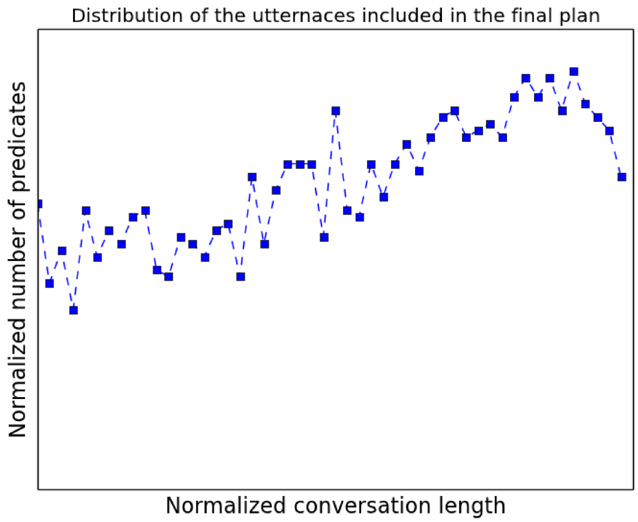


Figure 8: The distribution of the utterances included in the final plan (normalized)

However, a simple extension of the current model can relax this assumption and incorporate the different importance of each utterance. One way to decide an utterance's importance is to integrate human cognitive models. Human cognitive architectures (Anderson, 1983) model human cognitive operations, such as the memory model (Anderson, Bothell, Lebiere, & Matessa, 1998). For example, we can decrease the importance of each utterance as time proceeds in the planning session by applying varying weights to each utterance. A simple extension to the current model can be made to incorporate the memory model. Specifically, the variables ω_p and β can be modified to be vectors that represent weight or activation level of each utterance from the human cognition model (Anderson et al., 1998). The vector of ω_p will have the length of the utterances, $\omega_p = \{\omega_{p,1}, \omega_{p,2}, \cdots, \omega_{p,T}\}$, where each $\omega_{p,t}$ represents the activation level of each utterance. Similarly, we can extend β to be a vector, where each β_t can represent how noisy each utterance is, weighing it accordingly. However, while these cognitive models are empirically well-verified, Whitehill (2013) pointed out that there is no structured way to set parameters for these models. In addition, it is unclear how the human memory model would differ depending on the characteristics of a given task. For example, the memory model may differ significantly for short, succinct conversations conducted under time pressure.

7.5 Engineering the Proposal Distribution in the Metropolis-Hastings Sampling Algorithm

This section describes the impact of different proposal distributions in the MH sampling step, and our rationale for designing the proposal distribution as described in Section 4.3.1.

There have been numerous studies conducted on selecting a family of candidate-generating density functions (Metropolis et al., 1953; Hastings, 1970; Geweke, 1989; Gelman & Rubin,

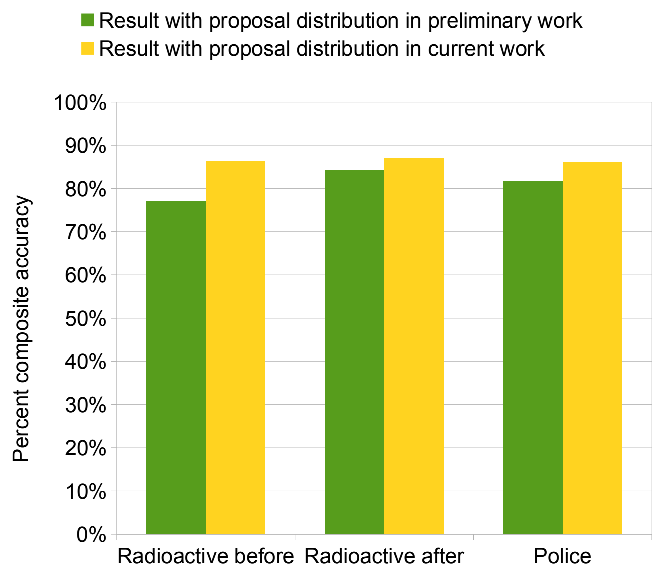


Figure 9: The impact of different proposal distributions (The highest accuracy with perfect PDDL files)

1992). However, as pointed out by Chib and Greenberg (1995), there is no structured way to choose the proposal distribution. It becomes more challenging when the sampled object is not a simple scalar variable, but a more complicated object, such as the *plan* variable (i.e., tuples of sets of grounded predicates) in this work. For such an object, there are larger spaces of potential proposal distributions to choose from.

However, a good choice for proposal distribution can improve performance. Figure 9 shows the results of the two different proposal distributions used for this work. The preliminary version of this work (Kim et al., 2013) applied the following distribution:

- Select a predicate from the set of possible predicates. If it is in the current *plan*, move it to either: 1) the next set of predicates or 2) the previous set, or 3) remove it from the current *plan*. If it is not in the current *plan*, move it to one of the existing sets.
- Select two sets in the current *plan* and switch their orders.

One difference between the proposal distribution above and the one outlined in Section 4.3.1 is the set of allowed timestamps that a selected predicate can move to at each iteration. The above proposed distribution allows a predicate to move to any timestamp, whereas the one in Section 4.3.1 only allows a predicate to move to an adjacent timestamp.

The key insight into proposal distribution in this work is gained by investigating sequences of MH sampling steps and observing when a proposal distribution fails to propose a good move. In other words, we identify what moves are necessary to move a proposed value (i.e., proposed new plan) to the true value of the latent variable (i.e., true plan) when they are close to each other. Often, a predicate is one timestamp off from the true timestamp (i.e., one timestamp after or before), and the proposal distribution as contained in the preliminary work (Kim et al., 2013) often fails to suggest a better proposed point. This motivated us to create a proposal distribution enabling more frequent moves between adjacent timestamps than between any two timestamps. As a result, we observed a substantial improvement to accuracy in all scenarios, as shown in Figure 9.

While this particular proposal distribution cannot be applied to all cases, this insight suggests that the following approach could be useful when designing a proposal distribution for non-scalar valued variables: First, a distance metric is defined between the two non-scalar valued variables. In our case, this step included defining the distance between two tuples of sets of predicates (i.e., *plan* variables). For example, the distance could be the average number of missing or extraneous predicates or the number of predicates that have incorrect timestamps. Second, starting from an initial proposed distribution, the distance between each sample and the true value is measured. Third, we can filter sample sequences when the distance is short, and visualize them. The shorter distance indicates moments when sampling could have almost reached the true value, but did not. Finally, the proposed distribution is modified to include the move that converts the samples in the third step to the true value within one or two moves. This process allows for insight into designing the proposal distribution. We leave further investigation of the systematic approach to future work.

8. Conclusion and Future Work

In this work, we have formulated the novel problem of performing inference to extract a finally agreed-upon plan from a human team's planning conversation. We presented an algorithm that combines a probabilistic approach with logical plan validation, used to compute a highly structured prior over possible plans. Our approach infers team plans without the need for historical data, using only situational information and data from a single planning session. We do not require the development or addition of a plan library to infer the plan, and demonstrate our solution is robust to incomplete knowledge of the planning problem. We demonstrated the benefit of this approach using human team meeting data collected from large-scale human subject experiments (total 96 subjects) and were able to infer the human teams' final plans with 86% accuracy on average.

In the future, we plan to build on this work to design an interactive agent that participates to improve human teams' planning decisions. Specifically we envision the work described here as a starting point for utilizing and building on human domain experts' knowledge, and improving the quality of finally agreed-upon plan through human-machine interaction.

9. Acknowledgement

This work is sponsored by ASD (R&E) under Air Force Contract FA8721-05-C-0002. Opinions, interpretations, conclusions and recommendations are those of the authors and are not necessarily endorsed by the United States Government.

Appendix A. The Visualization of Gibbs Sampling Convergence: Trace Plot

It is known that there is no conclusive way to determine whether the Markov chain of the Gibbs sampling has reached its stationary, or the desired posterior, distribution (Cowles & Carlin, 1996). Many available diagnostic tools are designed to test for necessary but insufficient conditions for convergence, such as work done by Gelman and Rubin (1992), Geweke (1991), Heidelberger and Welch (1981) and Raftery and Lewis (1995), to mention a few. In this work we utilize a much simpler yet still informative approach, which is to visually check whether convergence has been reached using the trace plot.

A trace plot is simply a scatter plot of the statistics of successive parameter estimates (e.g., the estimated values) with respect to the iteration steps. These statistics can be means, variances or covariance. A trace plot is most informative when the scalar variables are plotted. Figure 10 shows examples trace plots for the β and ω_p variables.

References

Albrecht, D. W., Zuckerman, I., Nicholson, A. E., & Bud, A. (1997). Towards a Bayesian model for keyhole plan recognition in large domains. In *Proceedings of the Sixth International Conference on User Modeling*, pp. 365–376. Springer-Verlag.

Anderson, J. R. (1983). A spreading activation theory of memory. *Journal of Verbal Learning and Verbal Behavior*, *22*(3), 261–295.

Anderson, J. R., Bothell, D., Lebiere, C., & Matessa, M. (1998). An integrated theory of list memory. *Journal of Memory and Language*, *38*(4), 341–380.

Andrieu, C., De Freitas, N., Doucet, A., & Jordan, M. I. (2003). An introduction to mcmc for machine learning. *Machine learning*, *50*(1-2), 5–43.

Barnes, M., Chen, J., Jentsch, F., & Redden, E. (2011). Designing effective soldier-robot teams in complex environments: training, interfaces, and individual differences. *EPCE*, 484–493.

Bauer, M., Biundo, S., Dengler, D., Koehler, J., & Paul, G. (2011). PHI: a logic-based tool for intelligent help systems..

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of Machine Learning Research*, *3*, 993–1022.

Brown, J. S., & Burton, R. R. (1978). Diagnostic models for procedural bugs in basic mathematical skills. *Cognitive Science*, *2*(2), 155–192.

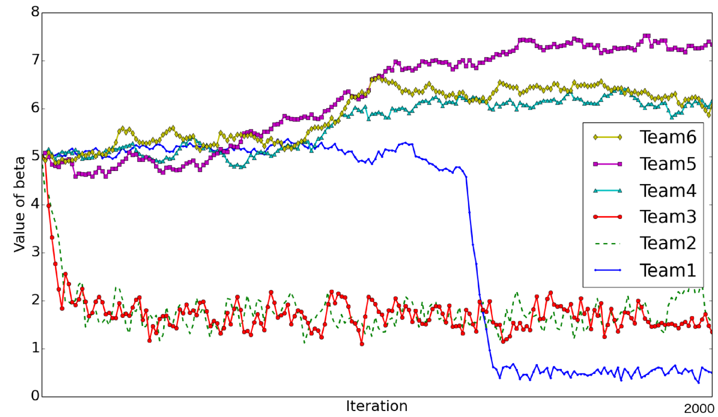
Carberry, S. (1990). *Plan recognition in natural language dialogue.* The MIT Press.

Casper, J., & Murphy, R. (2003). Human-robot interactions during the robot-assisted urban search and rescue response at the World Trade Center. *IEEE SMCS*, *33*(3), 367–385.

Casper, J., & Murphy, R. (2002). Workflow study on human-robot interaction in USAR. *IEEE ICRA*, *2*, 1997–2003.

Charniak, E., & Goldman, R. P. (1993). A Bayesian model of plan recognition. *Artificial Intelligence*, *64*(1), 53–79.

(a) Examples of β value convergences



(b) Examples of ω_p value convergence

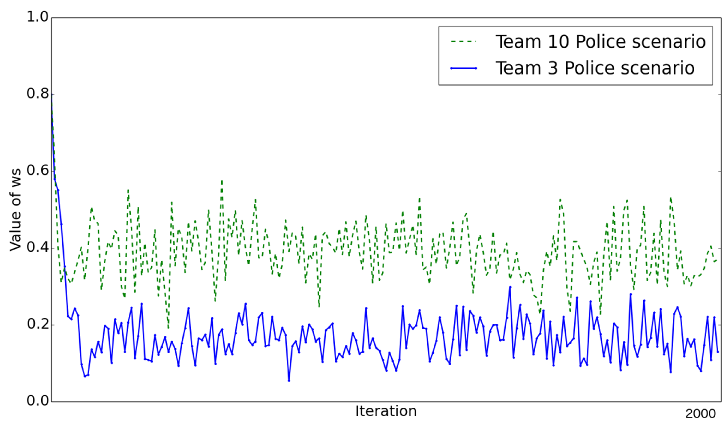


Figure 10: Trace Plots (only showing a subset of the data set)

Chib, S., & Greenberg, E. (1995). Understanding the Metropolis-Hastings algorithm. *The American Statistician*, *49*(4), 327–335.

Coles, A., Fox, M., Halsey, K., Long, D., & Smith, A. (2009). Managing concurrency in temporal planning using planner-scheduler interaction. *Artificial Intelligence*, *173*(1), 1–44.

Cowles, M. K., & Carlin, B. P. (1996). Markov chain Monte Carlo convergence diagnostics: a comparative review. *Journal of the American Statistical Association*, *91*(434), 883–904.

Cummings, M. L., Brzezinski, A. S., & Lee, J. D. (2007). Operator performance and intelligent aiding in unmanned aerial vehicle scheduling. *IEEE Intelligent Systems*, *22*(2), 52–59.

Di Ciaccio, R., Pullen, J., & Breimyer, P. (2011). Enabling distributed command and control with standards-based geospatial collaboration. *IEEE International Conference on HST*.

FEMA (2014). Federal emergency management agency.. [Online; accessed 3-December-2014].

Fikes, R. E., & Nilsson, N. J. (1972). Strips: A new approach to the application of theorem proving to problem solving. *Artificial intelligence*, *2*(3), 189–208.

Gal, Y., Reddy, S., Shieber, S. M., Rubin, A., & Grosz, B. J. (2012). Plan recognition in exploratory domains. *Artificial Intelligence*, *176*(1), 2270–2290.

Geib, C. W., & Goldman, R. P. (2009). A probabilistic plan recognition algorithm based on plan tree grammars. *Artificial Intelligence*, *173*(11), 1101–1132.

Geib, C. W., Maraist, J., & Goldman, R. P. (2008). A new probabilistic plan recognition algorithm based on string rewriting.. In *ICAPS*, pp. 91–98.

Gelman, A., & Rubin, D. B. (1992). Inference from iterative simulation using multiple sequences. *Statistical Science*, 457–472.

Getoor, L., & Mihalkova, L. (2011). Learning statistical models from relational data. *International Conference on Management of Data*, 1195–1198.

Geweke, J. (1989). Bayesian inference in econometric models using Monte Carlo integration. *Econometrica: Journal of the Econometric Society*, 1317–1339.

Geweke, J. (1991). *Evaluating the accuracy of sampling-based approaches to the calculation of posterior moments*. Federal Reserve Bank of Minneapolis, Research Department.

Goodrich, M. A., Morse, B. S., Engh, C., Cooper, J. L., & Adams, J. A. (2009). Towards using UAVs in wilderness search and rescue: Lessons from field trials. *Interaction Studies, Special Issue on Robots in the Wild: Exploring Human-Robot Interaction in Naturalistic Environments*, *10*(3), 453–478.

Grosz, B. J., & Sidner, C. L. (1990). Plans for discourse. In Cohen, P. R., Morgan, J., & Pollack, M. E. (Eds.), *Intentions in Communication*, pp. 417–444. MIT Press, Cambridge, MA.

Hastings, W. K. (1970). Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, *57*(1), 97–109.

Heidelberger, P., & Welch, P. D. (1981). A spectral method for confidence interval generation and run length control in simulations. *Communications of the ACM*, *24*(4), 233–245.

Holovaty, A., & Kaplan-Moss, J. (2009). *The definitive guide to Django: Web development done right*. Apress.

Horvitz, E., Breese, J., Heckerman, D., Hovel, D., & Rommelse, K. (1998). The Lumiere project: Bayesian user modeling for inferring the goals and needs of software users. *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, 256–265.

Howey, R., Long, D., & Fox, M. (2004). Val: Automatic plan validation, continuous effects and mixed initiative planning using PDDL. *IEEE ICTAI*, 294–301.

Jones, H., Rock, S., Burns, D., & Morris, S. (2002). Autonomous robots in SWAT applications: Research, design, and operations challenges. *AUVSI*.

Jurafsky, D., & Martin, J. H. (2000). *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition* (1st edition). Prentice Hall PTR, Upper Saddle River, NJ, USA.

Kautz, H. A., Pelavin, R. N., Tenenberg, J. D., & Kaufmann, M. (1991). A formal theory of plan recognition and its implementation. *Reasoning about Plans*, 69–125.

Kautz, H. A. (1987). *A formal theory of plan recognition*. Ph.D. thesis, Bell Laboratories.

Kim, B., Chacha, C. M., & Shah, J. (2013). Inferring robot task plans from human team meetings: A generative modeling approach with logic-based prior. *AAAI*.

Kim, J., & Shah, J. A. (2014). Automatic prediction of consistency among team members' understanding of group decisions in meetings. In *Systems, Man and Cybernetics (SMC), 2014 IEEE International Conference on*, pp. 3702–3708. IEEE.

Koomen, P., Punyakanok, V., Roth, D., & Yih, W. (2005). Generalized inference with multiple semantic role labeling systems. *CoNLL*, 181–184.

Kruijff, G., Janıcek, M., & Lison, P. (2010). Continual processing of situated dialogue in human-robot collaborative activities. In *IEEE Ro-Man*.

Larochelle, B., Kruijff, G., Smets, N., Mioch, T., & Groenewegen, P. (2011). Establishing human situation awareness using a multi-modal operator control unit in an urban search & rescue human-robot team. *IEEE Ro-Man*, 229–234.

Lochbaum, K. E. (1998). A collaborative planning model of intentional structure. *Computational Linguistics*, *24*(4), 525–572.

Mayfield, J. (1992). Controlling inference in plan recognition. *User Modeling and User-Adapted Interaction*, *2*(1-2), 55–82.

McDermott, D., Ghallab, M., Howe, A., Knoblock, C., Ram, A., Veloso, M., Weld, D., & Wilkins, D. (1998). PDDL-the planning domain definition language..

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., & Teller, E. (1953). Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, *21*, 1087.

Micire, M. (2002). *Analysis of robotic platforms used at the World Trade Center disaster*. Ph.D. thesis, MS thesis, Department Computer Science Engineering, Univ. South Florida.

Murphy, R. (2004). Human-robot interaction in rescue robotics. *IEEE SMCS*, *34*(2), 138–153.

Neal, R. M. (2003). Slice sampling. *Annals of Statistics*, 705–741.

Nguyen, T. A., Kambhampati, S., & Do, M. (2013). Synthesizing robust plans under incomplete domain models. *Advances in Neural Information Processing Systems*, 2472–2480.

Palmer, M., Gildea, D., & Xue, N. (2010). Semantic role labeling. *Synthesis Lectures on Human Language Technologies*, *3*(1), 1–103.

Pednault, E. P. D. (1987). Formulating Multi-Agent Dynamic-World Problems in the Classical Planning Framework. In *Reasoning About Actions and Plans: Proceedings of the 1986 Workshop*. Morgan Kaufmann Publishers.

Poon, H., & Domingos, P. (2006). Sound and efficient inference with probabilistic and deterministic dependencies. *AAAI*, *21*(1), 458.

Poon, H., & Domingos, P. (2009). Unsupervised semantic parsing. *EMNLP*.

Pradhan, S., Ward, W., Hacioglu, K., Martin, J., & Jurafsky, D. (2004). Shallow semantic parsing using support vector machines. *NAACL-HLT*, 233.

Pynadath, D. V., & Wellman, M. P. (2000). Probabilistic state-dependent grammars for plan recognition. *Proceedings of the Sixteenth conference on Uncertainty in Artificial Intelligence*, 507–514.

Raedt, L. (2008). Probabilistic logic learning. *Logical and Relational Learning*, 223–288.

Raftery, A. E., & Lewis, S. M. (1995). The number of iterations, convergence diagnostics and generic metropolis algorithms. *In Practical Markov Chain Monte Carlo*, 115–130.

Ramırez, M., & Geffner, H. (2009). Plan recognition as planning. *Proceedings of the 21st international joint conference on Artificial Intelligence*, 1778–1783.

Richardson, M., & Domingos, P. (2006). Markov logic networks. *Machine learning*, *62*(1), 107–136.

Ryall, K., Marks, J., & Shieber, S. (1997). An interactive constraint-based system for drawing graphs. *Proceedings of the 10th Annual ACM Symposium on User Interface Software and Technology*, 97–104.

Sadilek, A., & Kautz, H. A. (2010). Recognizing multi-agent activities from GPS data. *AAAI*.

Singla, P., & Domingos, P. (2007). Markov logic in infinite domains. *UAI*, 368–375.

Tellex, S., Kollar, T., Dickerson, S., Walter, M., Banerjee, A., Teller, S., & Roy, N. (2011). Understanding natural language commands for robotic navigation and mobile manipulation. *AAAI*.

Weida, R., & Litman, D. (1992). *Terminological Reasoning with Constraint Networks and an Application to Plan Recognition*.

Whitehill, J. (2013). Understanding ACT-R - an outsider's perspective. *CoRR*, *1306.0125*.

Zhuo, H. H., Yang, Q., & Kambhampati, S. (2012). Action-model based multi-agent plan recognition. *Advances in Neural Information Processing Systems 25*, 377–385.