GAPRec 2011

Proceedings of the 1st Workshop on Goal, Activity and Plan Recognition

ICAPS 2011

Freiburg, Germany – 12th June, 2011

Edited by

David Pattison, Derek Long and Christopher Geib

Organising Committee

Program Committee

Participating Authors

Foreword

Plan Recognition and its many variants represent an ever-increasingly important field of research. With the advent of intelligent agents who can reason for themselves, it becomes even more important to be able to reason about *other* agents.

In previous years the link between ICAPS and PR has been declining to the point where, in 2010, no recognition literature was presented. GAPRec represents the first time the recognition community have had an explicit presence at ICAPS and further offers a new avenue for upcoming recognition research. We are hopeful that this has invigorated those who attended and perhaps the wider planning community to submit their work to ICAPS in future years and potentially re-examine the unexplored links between planning and recognition.

One of the key aims of the workshop was to have an open and frank discussion on the possibility of a plan recognition competition. A summary of the outcome of this debate is included in these proceedings, with the consensus being that such a competition should undoubtedly happen in the near future. A competition would present an opportunity to overcome one of the major criticisms of recognition-related work – the lack of standardised evaluation schema and metrics. This in itself is often a barrier to acceptance at ICAPS and other conferences.

The workshop attracted a diverse set of submissions, which when combined represent an excellent snapshot of PR today and the current state-of-the-art. The organising committee wish to thank all those who submitted and presented their work, and also to those who attended the workshop for making it a success.

David Pattison, Derek Long and Christopher Geib.

A Plan Recognition Competition – Are We Ready?

David Pattison[1], Derek Long[1] and Christopher W. Geib[2]

[1]{david.pattison, derek.long}@cis.strath.ac.uk , Department of Computer and Information Science , University of Strathclyde , Glasgow G1 1XH, UK
[2]cgeib@inf.ed.ac.uk, School of Informatics , University of Edinburgh , Edinburgh EH8 9AB, UK

July 29, 2011

1 Introduction

Recently, there has been renewed interest and discussion within the Plan Recognition community as to the creation of a set of standard benchmarks against which researchers can evaluate their work. Further to this, the possibility of a PR competition has been proposed as a means of uniting the community and providing other methods of common evaluation.

This report summarises the opinions and viewpoints raised on these subjects in an open panel session held at the GAPRec workshop at ICAPS 2011. While the discussion held was about a PR competition as a whole, this report endeavours to extract the major topics, many of which would need to be resolved prior to a PR competition being held.

2 Competition Domains

The first hurdle to overcome in the creation of a PR competition is the problem set which recognisers would be evaluated against. Perhaps this is itself a worthy end-goal, as PR has historically had no access to a set of public, standardised problems on which to evaluate systems. This leads to each researcher developing their own, custom test suite for evaluation. The result of this is that it can be difficult to gain acceptance of PR work at major conferences, where a rigorous comparison with previous work is expected. With most work relying on the presence of plan libraries, many in the workshop were concerned at the scale of the problem in generating several of these for test problems. Many existing libraries model various aspects of the PR problem according to the author's specific research focus, such that it may be difficult to encourage participation from the majority of the community if the libraries used are too bespoke. To begin with, it may be best to produce a set of simple plan libraries which contain only HTN-style plans and simple probabilities.

Further to this, there was some discussion on whether a competition should be immediately separated into library-based and non-library based tracks. For the latter, it was suggested that existing IPC results could be used as a readily-available source of domains and plans, although there was a debate as to how plans which have been generated by a heuristic can be recognised in an unbiased manner. Tools currently exist for converting HTNs into flat, PDDL-esque domains, which may offer the ability for both tracks to run on the same domains (or even be evaluated against each other).

The complexity and difficulty of the problem set was also debated. Some felt that there should be a "simple" problem domain, and in contrast, a particularly "hard" domain, with any unsolved domains returning for successor competitions. However, quite how difficult the "hard" domain would be was undecided, although it was felt the complexity should come from the domain structure rather than from additional PR components such as multiple agents, adversarial agents, partial observability etc.

Of particular interest to a PR competition is the availability of domains prior to the competition on which to train recognisers. The general feeling was that if this were to be the case, there must be at least one "blind" domain which entrants are not aware of before the competition. The Learning track of the IPC was used as an example wherein test domains are supplied to entrants before the competition, but that these merely represent a "representation" of the true, hidden problem. Quite how this scenario would work in the context of training a Bayesian Network for execution on a different domain is unclear.

3 Standard Input

While the generation of plan libraries and test problems is itself a difficult problem to overcome, the agreement upon a standardisation of these and other aspects of the PR process must also be resolved.

There was much debate as to how a standard problem definition language in the vein of PDDL would be constructed, and if indeed this is even required. Some felt that the standardisation of plan libraries, plan structure and problem input was an inevitability which the PR community had avoided for too long, while others felt that establishing this principle would ultimately lead to the exclusion of many in the community who do not have the resources to adapt their work to a new input standard.

It was also suggested that problem input could itself be a worthy competition format as natural language parsing has historically been the target of PR research. In this case, the parsing of a paragraph containing the problem would be all that entrants were given, with the goal being to extract the problem input or plan. However, many felt that while this would make for an interesting competition track, it could detract from the main competition focus.

4 Evaluation

Agreement on a metric for a competition poses several challenges, not least of which is that the term *Plan Recognition* encompasses Plan Recognition, Goal Recognition, Activity Recognition, and all other variants. Given that each of these is often subtly different to the rest means that a common metric is probably impossible. Therefore, it was proposed that, in keeping with the spirit of simplicity, there could be 3 hypotheses

on each competition track.

1. **Next Action** – Prediction of the next action the agent will execute.
2. **Goal** – What the final goal will be.
3. **Plan** – What plan is being executed.

These 3 outputs are common to most recognisers, although the "Plan" output is perhaps limited to those doing fully-blown PR. With regard to scoring these outputs such that a ranking can be achieved, there was no consensus in the room. *Precision and Recall* has become a common metric in PR work and would satisfy the "Goal" output, but may not suit the "Plan" output. The "Next Action" output can simply be a binary score. A further metric of interest to the community is the processing time required to output a hypothesis, which can be used in all 3 of the above criteria.

5 Participation

As it stands today, one of the biggest challenges to the PR community is finding a single outlet for their research. Given that there is no home conference, work can be divergent across several conferences and disciplines, with only 2 or 3 workshops available dedicated to PR. While a competition would hopefully unite researchers, many in the workshop also saw it as an opportunity to attract entrants from other "non-traditional" PR disciplines.

Given the simple scoring metrics outlined above, entrants from fields such as robotics, natural-language processing and fault diagnosis could participate using their own research. However, many were quick to point out that many applications of PR are linked to military or private funding, and that these researchers may be excluded if their code had to be made public. It was unclear whether the source code for entrants should be made available to competition organisers.

Finally, a straw-poll was taken regarding how many of the GAPRec participants would be interested in participating in any future PR competition. Five people expressed an interest, which is comparable with the number of teams present in the first IPC in 1998, of which the most recent 2011 competition had 27 entrants.

6 Location

Opinions on where a PR competition should be held were divergent. Some believed that ICAPS would not be a suitable home as there has traditionally been only a small PR presence, while others felt that running alongside the IPC could help increase participation and foster interest in the competition. Other locations such as IJCAI or AAAI were suggested, as well as having the competition move about from year-to-year in the same manner as the PAIR workshop (which was also suggested as a suitable home).

7 Conclusions

The goal of the workshop was to determine whether any interest in a PR competition existed within the community, and to discuss what benefits and obstacles lay in the way of it becoming a reality. In this regard it was a success, with lively debate and opinions

given on all sides. Of course, not all matters were fully agreed upon, and these "grey-areas" would need to be the first things resolved in the creation of any competition. Even with a competition put to the side, the consensus on needing a common test-suite for PR evaluation was overwhelming – so much so that this may be the best course of action in the near future.

Contents

Corpus-Based Incremental Intention Recognition via Bayesian Network Model Construction

Han The Anh * and **Luís Moniz Pereira**
Centro de Inteligência Artificial (CENTRIA)
Departamento de Informática, Faculdade de Ciências e Tecnologia
Universidade Nova de Lisboa, 2829-516 Caparica, Portugal
$h.anh@fct.unl.pt, lmp@di.fct.unl.pt$

Abstract

We present a method for incremental intention recognition by means of incrementally constructing a Bayesian Network (BN) model as more actions are observed. It is achieved based on a knowledge base of easily maintained and constructed fragments of BNs, connecting intentions to actions. The simple structure of the fragments enables to easily and efficiently acquire the knowledge base, either from domain experts or automatically from a plan corpus. We show experimental results improvement for the Linux Plan Corpus. In addition, we create a new, so-called IPD Plan Corpus, for strategies in the iterated Prisoner's Dilemma and show the experimental results for it.

1. Introduction

We propose a method for intention recognition in a dynamic, real-world environment. An important aspect of intentions is their pointing to the future, i.e. if we intend something now, we mean to execute a course of actions to achieve something in the future (Bratman 1987). Most actions may be executed only at a far distance in time. During that period, the world is changing, and the initial intention may be changed to a more appropriate one or even abandoned (Bratman 1992). An intention recognition method should take into account these changes, and may need to reevaluate the intention recognition model depending on some time limit; in addition, as a new action is observed, the model should be reconfigurable to incorporate new observed actions.

Generally, *intention recognition* (also called *goal recognition*) is defined as the process of becoming aware of the intention of another agent and, more technically, as the problem of inferring an agent's intention through its actions and their effects on the environment (Heinze 2003). *Plan recognition* is closely related to intention recognition, extending it to also recognize the plan the observed agent is following in order to achieve his intention (Sadri 2010).

Intention recognition is performed in domains in which it is preferable to have a fast detection of just the user goal or intention than a more precise but time consuming detection of the complete user plan, e.g. in the interface agents domain (Horvitz et al. 1998). Generally, the input of both intention and plan recognition systems are a set of conceivable intentions and a set of plans achieving each intention (plan library or plan corpus). Intention recognition is distinct from planning, as goals are not known a priori, and presumed goals are subject to defeasibility. There are also generative approaches based on planning algorithms, which do not require plan library/corpus (e.g., see (Ramírez and Geffner 2010)).

In this work, we use Bayesian Networks (BN) as the intention recognition model. The flexibility of BNs for representing probabilistic dependencies and the efficiency of inference methods for BN have made them an extremely powerful and natural tool for problem solving under uncertainty (Pearl 1988; Pearl 2000).

This paper presents a knowledge representation method to support incremental BN construction for performing intention recognition during runtime, from an initially given domain knowledge base. As more actions are observed, a new BN is constructed reinforcing some intentions while ruling out others. This incremental method allows domain experts to specify knowledge in terms of small and simple BN fragments, which can be easily maintained and changed. Alternatively, these fragments can be easily learned from data.

It is inspired by the fact that knowledge experts often consider a related set of variables together, and organize domain knowledge in larger chunks. An ability to represent conceptually meaningful groupings of variables and their interrelationships facilitates both knowledge elicitation and knowledge base maintenance (Natarajan et al. 2008; Laskey 2008). To this end, there have been several methods proposed for Bayesian network construction from small and easily maintained network fragments (Pearl 1988; Natarajan et al. 2008; Laskey 2008). Basically, a combination of BNs is a graph that includes all nodes and links of the networks, where nodes with the same name are combined into a common node. The main issue for a combination method is how the influence of different parents of the common node can be combined in the new network, given the partial influence of each parent in the corresponding fragment. The most extensively used and popular combination method is Noisy-Or, firstly proposed by (Pearl 1988) for Bayesian networks of Boolean variables, and generalized by (Srinivas 1993) for the general case of arbitrary domains. A set of conditions is needed to be satisfied for the Noisy-OR method to work

*HTA is supported by FCT Portugal (reference SFRH/BD/62373/2009).

properly. We will discuss in more detail in the third section where the method is applied in our work (Def.5).

In the next section we recall some background about BN. Then a general method for incremental BN model construction during runtime is presented. We next address a special well-known case and present experimental results for it, comparing with other systems.

2. Bayesian Networks

Definition 1 *A Bayes Network is a pair consisting of a directed acyclic graph (DAG) whose nodes represent variables and missing edges encode conditional independencies between the variables, and an associated probability distribution satisfying the Markov assumption of conditional independence, saying that variables are independent of non-descendants given their parents in the graph (Pearl 2000).*

In a BN, associated with each node of its DAG is a specification of the distribution of its variable, say A, conditioned on its parents in the graph (denoted by $pa(A)$)—i.e., $P(A|pa(A))$ is specified. If $pa(A) = \emptyset$ (A is called root node), its unconditional probability distribution, $P(A)$, is specified. These distributions are called Conditional Probability Distribution (CPD) of the BN.

The joint distribution of all node values can be determined as the product of conditional probabilities of the value of each node on its parents $P(X_1, ..., X_N) = \prod_{i=1}^{N} P(X_i|pa(X_i))$, where $V = \{X_i|1 \leq i \leq N\}$ is the set of nodes of the DAG.

Suppose there is a set of evidence nodes (i.e. their values are observed) in the DAG, say $O = \{O_1, ..., O_m\} \subset V$. We can determine the conditional probability distribution of a variable X given the observed value of evidence nodes by using the conditional probability formula

$$P(X|O) = \frac{P(X,O)}{P(O)} = \frac{P(X,O_1,...,O_m)}{P(O_1,...,O_m)} \qquad (1)$$

where the numerator and denominator are computed by summing up the joint probabilities over all absent variables with respect to V.

3. Incremental Bayesian Network Construction for Intention Recognition

In (Pereira and Han 2009; Pereira and Han 2010), a general BN model for intention recognition is presented and justified based on Heinze's intentional model (Heinze 2003). Basically, the BN consists of three layers: cause/reason nodes in the first layer, connecting to intention nodes in the second one, in turn connecting to action nodes in the third. The interested readers are referred to those papers for more details. Han and Pereira (2010a) then presented a method for incrementally constructing such BN model for performing *incremental* intention recognition, including all the three layers.

A BN model for intention recognition consists of two layers: the layer of intentions and the layer of actions.

Definition 2 (Intention Recognition BN – IRBN) *A BN for intention recognition (IRBN) W is a triple $\langle\{Is, As\}, pa, P_W\rangle$ where*

- Is *and* As *are the sets of intention nodes and action nodes, respectively. They stand for binary random variables.*
- pa *is a mapping which maps a node to the set of its parent nodes such that: $\emptyset \neq pa(A) \subseteq Is \ \forall A \in As$, and $pa(I) = \emptyset \ \forall I \in Is$. This means there is no isolated action node and intentions are represented by top nodes.*
- *CPD tables of the action nodes and prior probabilities of the intention nodes are given by the probability distribution P_W, i.e. $P_W(X|pa(X))$ defines the probability of X conditional on $pa(X)$ in W, for all $X \in V_W$ where $V_W = Is \cup As$.*

The intention recognition method will be performed by incrementally constructing an IRBN as more actions are observed. The construction is based on a prior knowledge base of Unit BN Fragments consisting of a single intention connecting to a single action. We refer to them as the Unit Fragment (of BN) for intention recognition.

Definition 3 (Unit Fragment) *A Unit Fragment of BN for intention recognition consists of an intention I connecting to (i.e. causally affecting) an action A, and is denoted by $UF(I, A)$. Both nodes stand for binary random variables.*

Definition 4 (Knowledge Base) *A domain knowledge base KB consists of a set unit fragments.*

An intention I has the same *fixed* prior probability distribution in all the unit fragments it belongs to, denoted by $P_{KB}(I)$. The prior probability distribution of the top (intention) nodes also can be made situation-dependent by adding a pre-intentional layer of cause/reason nodes as in (Han and Pereira 2010a)—which would enable it to deal with and explain some important issues in intention/plan recognition such as intention change/abandonment (Geib and Goldman 2003). However, since the dataset we use later for evaluation has no such information available, we omit that layer to simplify the presentation.

The simple structure of unit fragments enables domain experts to easily construct and maintain the knowledge base. The BN fragments also can be learned from appropriate datasets, as we shall see later with the Linux plan corpus.

Before presenting the intention recognition algorithm, let us define some operators for handling CPD tables and IRBNs.

3.1 Operators for Constructing IRBN

As a new action A is observed, we need to incorporate it into the current IRBN. Firstly, the appropriate unit fragments for A are selected from the domain knowledge base. Han and Pereira (2010a) proposed some methods for selecting the appropriate fragments in a situation-sensitive manner. They are based on the intuition that whether an intention may give rise to an action depends on the situation in which the action is observed. That enables to reduce the size of the BN model, which otherwise could be very large.

We are not going to elaborate further on these methods here, and assume that the operator *select(A,SIT)* provides the (context-dependent) set of unit fragments for action A given the situation at hand SIT. If SIT is empty, *select(A,SIT)* is

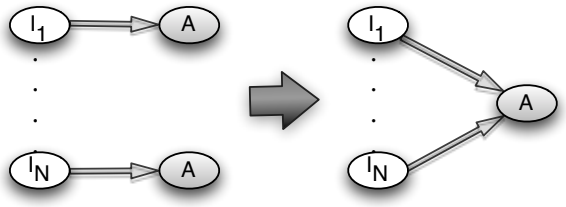


Figure 1: Noisy-OR Combination Method

the set of all unit fragments for action A from the knowledge base.

Secondly, after having obtained the appropriate fragments, we combine them using the Noisy-OR method (Pearl 1988; Srinivas 1993; Cozman 2004) and obtain an IRBN with a single action (Figure 1). It is called an Unit IRBN for action A in situation SIT, and denoted by *irBN(A,SIT)*.

Definition 5 (Unit IRBN via Noisy-OR) *The Unit IRBN for action* A *in a given situation* SIT, *$irBN(A, SIT)$, is obtained via* Noisy-OR *method as follows.*

Let $select(A, SIT) = \{UF(I_1, A), ..., UF(I_N, A)\}$ and for $1 \leq i \leq N$, $P(A = T|I_i = T) = p_i$ (defined in fragment $UF(I_i, A)$). Then, $\texttt{irBN(A, SIT)}$ *is the result of combining $UF(I_1, A), ..., UF(I_N, A)$ using Noisy-OR method, i.e. $p(A = T|I_1, ..., I_N) = 1 - \prod_{i:I_i=T}(1 - p_i)$. Note that the prior probability distribution of I_i, $1 \leq i \leq N$, in the new combined BN is the same as in its original fragment.*

The rationale and appropriateness of the application of the Noisy-OR method here for combining unit fragments is based on the intuition that each intention I_i, $1 \leq i \leq N$, can be interpreted as a "*cause*" of action A; and action A occurs when one or more of the intentions are active. Detailed arguments for this can be found in (Cozman 2004).

Definition 6 (Project of CPD Table) *Let T be a CPD table defining $P(X|V)$, the probability of a random variable X conditional on a set of random* binary *variables V, and $V' \subsetneq V$. The project of T on V', denoted by* $\texttt{proj(T, V')}$, *is the part of T corresponding to all variables in $V \setminus V'$ being false.*

Now we need to combine the obtained unit IRBN, $\texttt{irBN(A, SIT)}$, with the current IRBN. For that, in the sequel we define how to combine two IRBNs. Intuitively, we simply add up all the new nodes and links of the new IRBN to the current IRBN, keeping the CPD tables from the original IRBNs.

Definition 7 (Combination of IRBNs) *Let $W_1 = \langle\{Is_1, As_1\}, pa_1, P_1\rangle$ and $W_2 = \langle\{Is_2, As_2\}, pa_2, P_2\rangle$ be two IRBNs. The combination of these two IRBNs is an IRBN, denoted by* $\texttt{comb(W}_1\texttt{, W}_2\texttt{)} = \langle\{Is, As\}, pa, P_W\rangle$, *defined as follows*

- $As = As_1 \cup As_2$; $Is = Is_1 \cup Is_2$;
- $pa(I) = \emptyset$ $\forall I \in Is$; $pa(A) = pa_1(A) \cup pa_2(A)$;
- $P_W(I) = P_{KB}(I)$ $\forall I \in Is$; *and for each* $A \in As$, $P_W(A|pa(A)) = P_{W_1}(A|pa(A))$ *if* $A \in As_1$ *and* $P_W(A|pa(A)) = P_{W_2}(A|pa(A))$ *if* $A \in As_2$.

Note that here it is allowed the possibility that the observed agent follows multiple intentions simultaneously. In (Han and Pereira 2010a) the authors dealt with the case of a single intention being pursued, where in the combined IRBN only the intersection (instead of union) of two intention sets, $Is_1 \cap Is_1$, is retained; which enables to reduce the size of the IRBN model.

When some intentions are found irrelevant—e.g. because they are much unlikely[1]—those intentions should be remove from the IRBN. This is enacted by considering them as completely false and employing a project operator.

Definition 8 (Remove Intentions from IRBN) *Let $W = \langle\{Is, As\}, pa, P_W\rangle$ be an IRBN and $R \subset Is$ a strict subset of Is. The result of removing the set of intentions R from W is an IRBN, denoted by* $\texttt{remove(W, R)} = \langle\{Is_R, As_R\}, pa_R, P_R\rangle$, *and defined as follows*

- $As_R = \{A \in As \mid pa_R(A) \neq \emptyset\}$; $Is_R = Is \setminus R$;
- $pa_R(I) = \emptyset$ $\forall I \in Is_R$; $pa_R(A) = pa(A) \setminus R$;
- $P_W(I) = P_{KB}(I)$ $\forall I \in Is$; *and for each* $A \in As_R$, $P_R(A|pa_R(A))$ *is defined by the CPD table* $proj(T, Is_R)$ *where T is the CPD table for A in W, i.e. defined by* $P_W(A|pa(A))$.

Based on these operators, we now can describe an algorithm for incremental intention recognition in a real-time manner.

Incremental Intention Recognition Algorithm. Repeat the following step until some given time limit is reached; the most likely intention in previous cycle is the final result.

- Let A be a new observed action and SIT the current situation. Combine the current IRBN W with $\texttt{irBN(A, SIT)}$ we obtain $\texttt{W}' = \texttt{comb(W, irBN(A, SIT))}$. If A is the initially observed action, let $\texttt{W}' = \texttt{irBN(A, SIT)}$.
- Compute the probability of each intention in W', conditional on the set of current observed actions in W'. Remove the intentions which are much less likely than the others (following Definition 8).

Note that if an observed action is not in the knowledge base, the action is considered irrelevant to the sought for intention, and discarded. Furthermore, at any cycle, if the likelihood of all the intentions are very small (say, smaller than a given threshold), one could say that the sought for intention is abandoned. This is because the causes and actions do not support or force the intending agent to keep pursuing his initial intention anymore.

4. Relations Amongst Intention Nodes

When considering the case in which the observed agent may pursue multiple intentions simultaneously, it is undoubtedly indispensable to take into account and express the relations amongst the intentions in the model.

Pursuing one intention may exclude the other intention to be pursued. It may be so because of resource limitation, e.g.

[1] One intention is much less likely than the other if the fraction of its likelihood and that of the most likely intention is less than some small threshold. It is up to the KB designer to provide it.

allowance time is not enough for accomplishing both intentions at the same time. It also may be because of the nature or restriction of the observed agent's task: the agent is restricted to pursuing a single intention (e.g. in constructing Linux plan corpus, a user is given one task at a time to complete. We shall discuss this case in more detail in the next sections).

We introduce a so-called *exclusive relation e*—a binary relation on the set of intention nodes—representing that if one intention is pursued, then the other intention cannot be pursued. It is usually, although perhaps not always, the case that intentions exclusiveness is symmetric. It holds for the resource limitation case: one intention excludes the other intention because there is not enough resource for accomplishing both, which in turn implies that the latter intention also excludes the former one. It also clearly holds for the case where the agent is restricted to pursuing a single intention. In this paper, we assume that the exclusive relation on intentions e is symmetric; it can be renamed *mutually exclusive relation*.

Intentions I_1 and I_2 are mutually exclusive iff they cannot be pursued simultaneously, i.e. $P(I_1 = T, I_2 = T) = 0$. Thus, for any action A, if $I_1, I_2 \in pa(A)$ then the CPD table for A is undefined. Hence, the BN needs to be restructured. The mutually exclusive intentions must be combined into a single node since they cannot co-exist as parents of a node. Each intention represents a possible value of the new combined node. Namely, let $I_1, ..., I_t$ be such that $e(I_i, I_j),\ \forall i, j : 1 \leq i < j \leq t$. The new combined node, I, stands for a random variable whose possible outcomes are either $I_i,\ 1 \leq i \leq t$, or $\tilde{I}$—the outcome corresponding to the state that none of $I_i = T$. Note that if the intentions $I_1, ..., I_t$ are exhaustive, $\tilde{I}$ can be omitted. Next, I is linked to all the action nodes that has a link from one of $I_i,\ 1 \leq i \leq t$.

It remains to re-define CPD tables in the new BN. They are kept the same for action A where $I \notin pa(A)$. For A such that $I \in pa(A)$, the new CPD table at $I = I_i$ corresponds to the CPD table in the original BN at $I_i = T$ and $I_j = F\ \forall j \neq i, 1 \leq j \leq t$, i.e. $P(A|I = I_i, ...) = P(A|I_0 = F, ..., I_{i-1} = F, \mathbf{I_i = T}, I_{i+1} = F, ..., I_t = F, ...)$. Note that the left hand side is defined in the new BN, and the right hand side is defined in the original BN. Similarly, the new CPD table at $I = \tilde{I}$ corresponds to $I_i = F\ \forall 1 \leq i \leq t$. In addition, prior probability $P(I = I_i) = P(I_i = T)$ and $P(I = \tilde{I}) = \prod_{i=1}^{t} P(I_i = F)$ (and then being normalized).

In the next section we will look at a special case where the observed agent pursues a single intention. Thus, all intentions are mutually exclusive, and they can be combined into a single node. We then evaluate the method using the Linux Plan Corpus. After that, in Section 6., we present a new plan corpus, called IPD, and also present experimental results for it.

5. Single Intention Being Pursued

5.1 The Model

Suppose the observed agent pursues a single intention. In this case, all intentions are mutually exclusive, and they can be combined into a single node. The IRBN then consists of a single intention node, linking to all action nodes.

Let $I_1, ..., I_n$ be the intentions in the original IRBN. As usual, they are assumed to be exhaustive, i.e. the observed agent is assigned an intention from them. The combined node I thus has n possible outcomes $I_i,\ 1 \leq i \leq n$. Let $A_1, ..., A_m$ be the current observed actions. Applying Equation 1, we easily obtain the probability of each intention conditional on the current observed actions as follows, for $1 \leq j \leq n$,

$$P(I = I_j | A_1, ..., A_m) = \frac{P(I_j) \prod_{i=1}^{m} P(A_i|I_j)}{\sum_{j=1}^{n} P(I_j) \prod_{i=1}^{m} P(A_i|I_j)}$$

This implies our intention recognizer has a linear complexity $O(|Is|)$, where Is is the set of intentions being modeled.

5.2 Experimental Evaluation

The Linux Plan Corpus Plan corpus is the term used to describe a set of plan sessions and consists of a list of goals/intentions and the actions a user executed to achieve them (Armentano and Amandi 2009). Although there are many corpora available for testing machine learning algorithms in other domains, just a few are available for training and testing plan/intention recognizers; furthermore, each of the plan/intention recognizers using plan corpora usually has its own datasets—which leads to a difficult comparison amongst each other. For that important reason, we chose Linux plan corpus (Blaylock and Allen 2004)—one of the rare regularly used plan corpora—which was kindly made publicly available by Nate Blaylock—in order to test our system. It also enables a better comparison with other systems using this corpus (Blaylock and Allen 2005; Blaylock and Allen 2004; Armentano and Amandi 2009).

The Linux plan corpus is modeled after Lesh's Unix plan corpus (Lesh 1998). It was gathered from 56 human users (graduate and undergraduate students, faculty, and staff) from the University of Rochester Department of Computer Science. The users have different levels of expertise in the use of Linux, and they were allowed to perform as many times as they wished, in order to contribute more plan sessions. The sessions, consisting in sequences of commands performed by the users to achieve a given goal/intention, were automatically recorded. At the end of each session, the users were asked to indicate whether they succeeded in achieving their goal/intention. In total, there are 547 sessions, 457 of which were indicated as successfully completing the goal, 19 goal schemas and 43 action schemas. More details can be found in (Blaylock and Allen 2004) or (Linux-Plan-Corpus).

Learning Unit Fragments from Data For unit fragment $UF(I, A)$, the conditional probability of A given I is defined by the frequency of A in a plan session for achieving the goal/intention I divided by the frequency of any action for achieving I: $P(A = T|I = T) = \frac{freq(A_I)}{freq(I)}$. For better understanding, in the plan corpus each action is marked with the intention which the action is aiming at. Then, $freq(A_I)$ is the frequency of A being marked by I, and $freq(I)$ is the frequency of seeing the mark I.

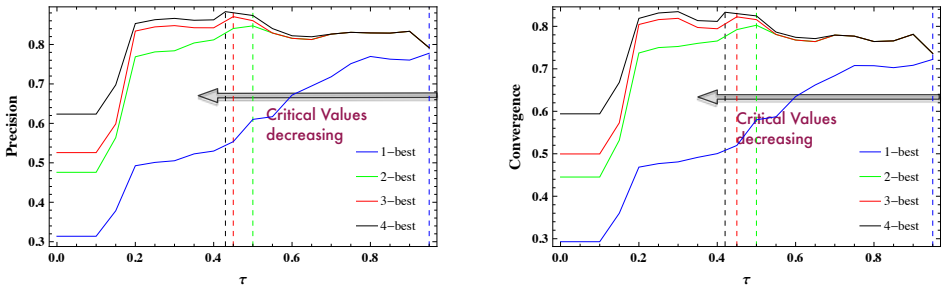


Figure 2: Precision and convergence for $\tau \in [0, 1]$ and for different values of N (N = 1, 2, 3, 4) on Linux Plan corpus.

Note that prior probabilities of all the intentions in the corpus are given initially, and used for generating tasks for users (Linux-Plan-Corpus ; Blaylock and Allen 2004).

Making Predictions Similar to (Blaylock and Allen 2004), instead of letting the recognizer make a prediction after each observed action, we set a *confidence* threshold τ $(0 \leq \tau \leq 1)$, which allows the recognizer to decide whether or not it is confident enough to make a prediction; the recognizer only makes a prediction if the likelihood of the most likely intention in the model is greater than τ. Otherwise, it predicts "don't know". In addition, instead of only predicting the most likely intention, the recognizer provides a set of N most likely ones (*N-best prediction*).

Evaluation Metrics For evaluating our system and comparing with the previous ones (Blaylock and Allen 2004; Armentano and Amandi 2009), we use three different metrics. *Precision* and *recall* report the number of correct predictions divided by total predictions and total prediction opportunities, respectively. More formally (also see (Armentano and Amandi 2009)), let $Seq = a_1, ..., a_n$ be a sequence of actions (plan session) achieving intention I. Considering N-best prediction case, let $correct(A) = 1$ if I is one of N most likely intentions, and 0 otherwise. Then, precision and recall for *Seq* are defined as: $precision(Seq) = (\sum_{i=1}^{n} correct(a_i))/z$; $recall(Seq) = (\sum_{i=1}^{n} correct(a_i))/Z$, where z and Z are the number of predictions made (when the recognizer is confident enough) and the total number of prediction opportunities (i.e. when $\tau = 0$), respectively.

On the other hand, *convergence* is a metric that indicates how much time the recognizer took to converge on what the current user goal/intention was. Let t be such that $correct_i = 0$ for $0 \leq i \leq t-1$ and 1 for $t \leq i \leq n$ (i.e. t is the first time point which from there on the system always correctly predicts), *convergence* for *Seq* is defined as: $convergence(Seq) = (z - t + 1)/z$.

Finally, the *overall* precision, recall and convergence are obtained by taking averages over all testing sessions.

Experiments and Results Because of the small size of the Linux corpus, similar to previous works, we ran experiments using the one-out cross validation method (Armentano and Amandi 2009). Just one at a time, one plan session in the whole corpus is left out. The rest of the corpus is used for training the model, which is then evaluated against the left out plan session. We study the effect of confidence level τ w.r.t. precision and convergence (for recall, it clearly is a decreasing function of τ) (Figure 2). The greater N, the better precision and convergence. The difference in precision and convergence between two different values of N is large when τ is small, and gets smaller for greater τ. Most interestingly, we observe that precision and convergence are not increasingly monotonic on τ. There are *critical values* of τ at which the measures have maximal value, and those values are smaller for greater N. This observation suggests that in plan/intention recognition task, the more precise (i.e. the smaller N) the decision is needed to make, the greater confidence level the recognizer should gain to make a good (enough) decision. On the other hand, the recognizer should not be too cautious, leading to refuse to make a prediction when it would have been able to make a correct one. In short, this experimentation suggests an important need to study (experimentally) the confidence threshold τ carefully for particular application domains, and for particular values of N. Using the same τ for all values of N could decrease the recognizer's performance.

Table 1: Intention Recognition Results on the Linux Plan Corpus

N-best	1-best	2-best	3-best	4-best
τ	0.95	0.5	0.45	0.42
Precision	0.786	0.847	0.870	0.883
Recall	0.308	0.469	0.518	0.612
Converg.	0.722	0.799	0.822	0.824

Table 1 shows some of the results for different values of N (and the corresponding value of τ). Similar to the previous works on the same Linux corpus (Blaylock and Allen 2004; Armentano and Amandi 2009), we keep the best results of each case w.r.t. τ for the comparison. For example, we obtained a precision of 78.6% for 1-best that is increased to 87.0% for 3-best prediction and 88.3% for 4-best one. Convergence is increased from 72.2% for 1-best to 82.2% for 3-best and 82.4% 4-best prediction.

The best performance on the Linux corpus (namely, in terms of precision and convergence) so far was reported in (Armentano and Amandi 2009), where the authors use variable Markov model with exponential moving average. Here we got an increment of 14% better precision and 13.3% better convergence for 1-best prediction, 8.2% better precision and 9.3% better convergence for 2-best prediction, and 7.5% better precision and 7.7% better convergence for 3-

best prediction. We also obtained better recalls comparing with (Blaylock and Allen 2004) in all cases.

Note that in (Armentano and Amandi 2009), the authors use a more fine-grained preprocessing method for their work, but we suspect it would have increased their performance. To fairly compare with both works, we use the original corpus.

6. IPD Plan Corpus

We present a new plan corpus in the context of Iterated Prisoner's Dilemma (IPD)[2] and show the experimental results for it. The intentions/goals to be recognized are the (known) strategies in IPD (see below) and plan sessions are the sequences of moves these strategies play with other players.

6.1 Iterated Prisoner's Dilemma

Prisoner's Dilemma is a symmetric two-player non-zero game defined by the payoff matrix (for row player)

$$\begin{array}{cc} & \begin{array}{cc} C & D \end{array} \\ \begin{array}{c} C \\ D \end{array} & \begin{pmatrix} R & S \\ T & P \end{pmatrix} \end{array}$$

Each player have two options in each round, cooperates (C) or defects (D). A player who chooses to cooperate with someone who defects receives the sucker's payoff S, whereas the defecting player gains the temptation to defect, T. Mutual cooperation (resp., defection) yields the reward R (resp., punishment P) for both players. In PD, it satisfies that $T > R > P > S$. Thus, in a single round, it is always best to defect, but cooperation may be rewarded if the game is iterated. In IPD, it is also required that mutual cooperation is preferred over an equal probability of unilateral cooperation and defection ($2R > T + S$); otherwise alternating between cooperation and defection would lead to a higher payoff than mutual cooperation.

IPD is usually known as a story of tit-for-tat (TFT), which won both Axelrod's tournaments (Axelrod 1984). *TFT* starts by cooperating, and does whatever the opponent did in the previous round. It will cooperate if the opponent cooperated, and will defect if the opponent defected. But if there are erroneous moves (i.e. an intended move is wrongly performed with a given execution error), the performance of *TFT* declines: it cannot correct errors or mistakes. Tit-for-tat is then replaced by generous tit-for-tat (GTFT), a strategy that cooperates if the opponent cooperated in the previous round, but sometimes cooperates even if the opponent defected (with a fixed "forgiveness" probability $p > 0$) (Sigmund 2010). *GTFT* can correct mistakes.

Subsequently, *TFT* and *GTFT* were replaced by win-stay-lose-shift (WSLS) as the winning strategy chosen by evolution (Sigmund 2010). *WSLS* repeats the previous move whenever it did well, but changes otherwise.

Some other less famous strategies (which we are going to use later) are GRIM – a grim version of TFT, prescribing to defect except after a round of mutual cooperation, and Firm-But-Fair (FBF) – known as a tolerant brother of TFT, prescribing to defect only if getting a sucker's payoff S in previous round. Details of all strategies considered here can be found in (Sigmund 2010) (Chapter 3).

Next, we describe how training and testing plan corpora are created employing these strategies. Abusing notations, R, S, T and P are also referred to as game states (in a single round or interaction). We too use E (standing for *empty*) to refer to the game state having had no interaction.

6.2 IPD Plan Corpus Description

We made an assumption that all strategies to be recognized have the memory size bounded-up by M ($M \geq 0$)—i.e. their decision at the current round is independent of the past rounds that are at a time distance greater than M. The strategies described above have memory $M = 1$.

An action in the corpus is of the form $s_1 ... s_M \xi$, where $s_i \in \{E, R, T, S, P\}$, $1 \leq i \leq M$, are the states of the M last interactions, and $\xi \in \{C, D\}$ is the current move. We denote by Σ_M the set of all possible types of action. E.g, $\Sigma_1 = \{EC, RC, TC, SC, PC, ED, RD, TD, SD, PD\}$. This encoding method enables to save the game states without having to save the co-player's moves, thus simplifying the corpus representation, described below.

Suppose we have a set of strategies to be recognized. The plan corpus for this set consists of a set of plan sessions generated for each strategy in the set. A plan session of a strategy is a sequence of actions played by that strategy (more precisely, a player using that strategy) against an arbitrary player. As an example, let us consider *TFT* and the following sequence of its interactions with some other player (denoted by X), in the presence of noise

round :	0	1	2	3	4	5
TFT :	–	C	C	D	D	D
X :	–	C	D	D	C	D
TFT-states :	E	R	S	P	T	P

The corresponding plan session for *TFT* is $[EC, RC, SD, PD, TD]$. At 0-th round, there is no interaction, thus the state is E. *TFT* starts by cooperating (1-st round), hence the first action of the plan session is EC. Since player X also cooperates in the 1-st round, the game state at this round is R. *TFT* reciprocates in the 2-nd round by cooperating, hence the second action of the plan session is RC. Similarly for the third and the fourth actions. Now, at the 5-th round, *TFT* should cooperate since X cooperated in 4-th round, but because of noise, it makes an error to defect. Therefore, the 5-th action is TD.

6.3 Plan Corpora Generation

Let us start by generating a plan corpus of seven strategies within the IPD framework: *AllC* (always cooperate), *AllD* (always defect), *TFT*, *GTFT* (probability of forgiveness a defect is $p = 0.5$), *WSLS*, GRIM and FBF.

We collect plan sessions of each strategy by playing a random choice (C or D) in each round with it. To be more thorough, we can also play all the possible combinations for each given number of rounds to be played. For example, if it is 10, there will be 1024 (2^{10}) combinations—C or D in each

[2] It also applies for other famous social dilemmas such as Snow Drift and Stag Hunt (Sigmund 2010).

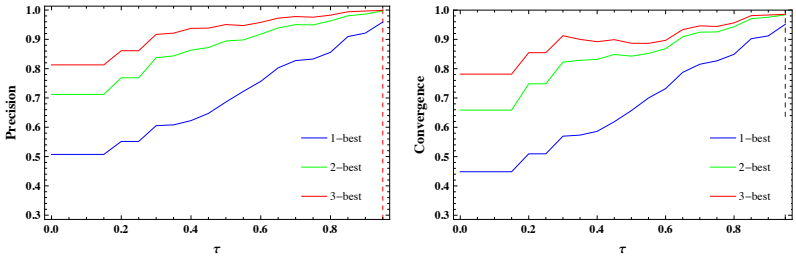


Figure 3: Plot of our method's precision and convergence for $\tau \in [0, 1]$ and for different values of N (N = 1, 2, 3) in IPD Plan Corpus.

round. When noise is present, each combination is played repeatedly several times.

The training corpus is generated by playing with each strategy all the possible combinations 10 times, and for each number of rounds from 5 to 10. The testing dataset is generated by playing a random choice with each strategy in each round, and also for each number of rounds from 5 to 10. We continue until obtaining the same number of sessions as in the training dataset (corpus). Both datasets are generated in presence of noise (namely, an intended move is wrongly performed with probability 0.05).

6.4 Results

The intention recognition model is acquired using the training corpus. Figure 3 shows the precision and convergence of the model with respect to the testing dataset. Given that the training as well as the testing datasets are generated in presence of noise, the achieved intention recognition performance is quite good. Namely, for big enough τ, both precision and convergence scores are greater than 0.9, even for the 1-best case.

7. Related Work

Bayesian networks have been one of the most successful models applied for intention/plan recognition problem (most importantly, see, (Charniak and Goldman 1993; Geib and Goldman 2009)). Depending on the structure of plan libraries, they employed some knowledge-based model construction to build BNs from the library, and then infer the posterior probability of explanations (for the set of observed actions). These works address a number of issues in intention/plan recognition, e.g. the observed agent follows multiple intentions or interleaved plans simultaneously; fails to observe actions; addresses partially ordered plans. However, they made several assumptions for the sake of computational efficiency. First, the prior probabilities of intentions are assumed to be fixed. This assumption is not reasonable because those prior probabilities should depend on the situation at hand, and be captured by the causes/reasons of intentions (see (Pereira and Han 2010) for several examples). In (Pynadath and Wellman 1995), a similar context-dependent approach was used, although the model is not incremental. Second, intentions are assumed to be independent of each other. This is not generally the case since the intentions may support or exclude one another. Those works hence do not appropriately address multiple intention recognition.

This latter assumption must always, explicitly or implicitly, be made by the approaches based on (Hidden) Markov Models, e.g. (Armentano and Amandi 2009; Bui 2003), or statistical corpus-based machine learning (Blaylock and Allen 2004; Blaylock and Allen 2005). Generally, in those approaches, a separate model is built for each intention; thus no relations amongst the intentions are expressed or can be expressed. These works were restricted to the single intention case.

8. Conclusion and Future Work

We have presented a method for incremental intention recognition. The method is performed by dynamically constructing a BN model for intention recognition from a prior domain knowledge base consisting of easily maintained fragments of BN. A fragment consists of a single intention connecting to a single action. This simple network structure allows easy maintenance by domain experts as well as automatically building from available plan corpora.

The main contribution of this paper is our efficient, yet very simple, method for incremental intention recognition. In general, its performance is better than all existent ones that make use of the Linux corpus. Given that our method also has the same linear complexity as other (best) existent ones, and that our model learning process is much simpler, we believe to have achieved significant improvements. In addition, the good performance of the method with respect to the Linux corpus shows its applicability to the important interface-agents domain (Horvitz et al. 1998).

The other contribution, though perhaps only minor, is our method for multiple intention recognition. We have proposed how to represent relationships amongst intentions in the intention recognition model. This aspect is indispensable in multiple intention recognition, but always omitted in previous works. Our next step is to evaluate the method experimentally. Note that although elsewhere reported a capability of dealing with the case where multiple intentions are being followed (e.g. (Geib and Goldman 2009)), to the best of our knowledge that capability has never been evaluated experimentally, partly due to unavailability of appropriate plan corpora or benchmarks. Thus, for the evaluation, we must gather an appropriate plan corpus allowing for the possibility that users pursue multiple intentions simultaneously.

In addition, for the intention recognition community, given the rich set of strategies in the literature (Hofbauer and Sigmund 1998; Sigmund 2010), we have provided here an important, easily extendable benchmark for evaluating intention recognition methods. Given that IPD and other social dilemmas are regularly found in everyday life, and the strategies studied within the framework of those dilem-

mas actually reflect human behaviors, we believe that game theory (and more generally, evolutionary game theory (Hofbauer and Sigmund 1998)) is a highly promising framework for creating benchmarks for intention recognition. Methods applicable for this benchmark can be used for a wide range of application domains, as diverse as in economics, psychology and biology (Sigmund 2010). For example, our intention recognition model has been successfully used to study the role of intention recognition in the evolution of cooperation, one of the most important issues actively studied in those fields (Han, Pereira, and Santos 2011; Han, Pereira, and Santos).

We also aim at a real deployment for some application domains such as Elder Care (Pereira and Han 2010) and Ambient Intelligence (Han and Pereira 2010b). The simple structure of our BN fragments would enable an easy data collection process.

References

[Armentano and Amandi 2009] Armentano, M. G., and Amandi, A. 2009. Goal recognition with variable-order Markov models. In *IJCAI'09*, 1635–1640.

[Axelrod 1984] Axelrod, R. 1984. *The Evolution of Cooperation*. Basic Books, ISBN 0-465-02122-2.

[Blaylock and Allen 2004] Blaylock, N., and Allen, J. 2004. Statistical goal parameter recognition. In *ICAPS04*, 297–304. AAAI.

[Blaylock and Allen 2005] Blaylock, N., and Allen, J. 2005. Recognizing instantiated goals using statistical methods. In *IJCAI Workshop on Modeling Others from Observations (MOO-2005)*, 79–86.

[Bratman 1987] Bratman, M. E. 1987. *Intention, Plans, and Practical Reason*. The David Hume Series, CSLI.

[Bratman 1992] Bratman, M. E. 1992. Planning and the stability of intention. *Minds and Machines* 2(1):1–16.

[Bui 2003] Bui, H. H. 2003. A general model for online probabilistic plan recognition. In *IJCAI'03*, 1309–1318.

[Charniak and Goldman 1993] Charniak, E., and Goldman, R. 1993. A Bayesian model of plan recognition. *Artificial Intelligence* 64(1):53–79.

[Cozman 2004] Cozman, F. G. 2004. Axiomatizing noisy-or. In *ECAI-04*, 979–980.

[Geib and Goldman 2003] Geib, C. W., and Goldman, R. P. 2003. Recognizing plan/goal abandonment. In *IJCAI'03*.

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

[Han and Pereira 2010a] Han, T. A., and Pereira, L. M. 2010a. Anytime intention recognition via incremental Bayesian network reconstruction. In *AAAI 2010 Fall Symp. on Proactive Assistant Agents*, 20–25. AAAI.

[Han and Pereira 2010b] Han, T. A., and Pereira, L. M. 2010b. Proactive intention recognition for home ambient intelligence. In *IE Workshop on AI Techniques for Ambient Intelligence*, 91–100. IOS Press.

[Han, Pereira, and Santos] Han, T. A.; Pereira, L. M.; and Santos, F. C. Intention recognition promotes the emergence of cooperation. *Adaptive Behavior*. to appear 2011.

[Han, Pereira, and Santos 2011] Han, T. A.; Pereira, L. M.; and Santos, F. C. 2011. The role of intention recognition in the evolution of cooperative behavior. In *IJCAI'2011*. to appear.

[Heinze 2003] Heinze, C. 2003. *Modeling Intention Recognition for Intelligent Agent Systems*. Ph.D. Dissertation, The University of Melbourne, Australia.

[Hofbauer and Sigmund 1998] Hofbauer, J., and Sigmund, K. 1998. *Evolutionary Games and Population Dynamics*. Cambridge University Press.

[Horvitz et al. 1998] Horvitz, E.; Breese, J.; Heckerman, D.; Hovel, D.; and Rommelse, K. 1998. The Lumiere project: Bayesian user modeling for inferring the goals and needs of software users. In *UAI'98*, 256–265.

[Laskey 2008] Laskey, K. B. 2008. Mebn: A language for first-order Bayesian knowledge bases. *Artificial Intelligence* 172(2-3):140 – 178.

[Lesh 1998] Lesh, N. 1998. *Scalable and Adaptive Goal Recognition*. Ph.D. Dissertation, U. of Washington.

[Linux-Plan-Corpus] Linux-Plan-Corpus. http://www.cs.rochester.edu/research/cisd/resources/linux-plan/. Last access: November 21 2010.

[Natarajan et al. 2008] Natarajan, S.; Tadepalli, P.; Dietterich, T. G.; and Fern, A. 2008. Learning first-order probabilistic models with combining rules. *Annals of Mathematics and Artificial Intelligence* 54:223–256.

[Pearl 1988] Pearl, J. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann.

[Pearl 2000] Pearl, J. 2000. *Causality: Models, Reasoning, and Inference*. Cambridge U.P.

[Pereira and Han 2009] Pereira, L. M., and Han, T. A. 2009. Intention recognition via causal Bayes networks plus plan generation. In *Procs. of 14th Portuguese Intl. Conf. on Artificial Intelligence (EPIA'09)*, 138–149. LNAI 5816.

[Pereira and Han 2010] Pereira, L. M., and Han, T. A. 2010. Intention recognition with evolution prospection and causal Bayesian networks. In *Computational Intelligence for Engineering Systems*. Springer. 1–33.

[Pynadath and Wellman 1995] Pynadath, D. V., and Wellman, M. P. 1995. Accounting for context in plan recognition, with application to traffic monitoring. In *UAI*.

[Ramírez and Geffner 2010] Ramírez, M., and Geffner, H. 2010. Probabilistic plan recognition using off-the-shelf classical planners. In *AAAI'2010*.

[Sadri 2010] Sadri, F. 2010. Logic-based approaches to intention recognition. In *Handbook of Research on Ambient Intelligence: Trends and Perspectives*.

[Sigmund 2010] Sigmund, K. 2010. *The Calculus of Selfishness*. Princeton U. Press.

[Srinivas 1993] Srinivas, S. 1993. A generalization of the noisy-or model. In *UAI'93*.

Goal Recognition over POMDPs: Inferring the Intention of a POMDP Agent*

Miquel Ramírez
Universitat Pompeu Fabra
08018 Barcelona, SPAIN
miquel.ramirez@upf.edu

Hector Geffner
ICREA & Universitat Pompeu Fabra
08018 Barcelona, SPAIN
hector.geffner@upf.edu

Abstract

Plan recognition is the problem of inferring the goals and plans of an agent from partial observations of her behavior. Recently, it has been shown that the problem can be formulated and solved using planners, reducing plan recognition to plan generation. In this work, we extend this model-based approach to plan recognition to the POMDP setting, where actions are stochastic and states are partially observable. The task is to infer a probability distribution over the possible goals of an agent whose behavior results from a POMDP model. The POMDP model is shared between agent and observer except for the true goal of the agent that is hidden to the observer. The observations are action sequences O that may contain gaps as some or even most of the actions done by the agent may not be observed. We show that the posterior goal distribution $P(G|O)$ can be computed from the value function $V_G(b)$ over beliefs b generated by the POMDP planner for each possible goal G. Some extensions of the basic framework are discussed, and a number of experiments are reported.

Introduction

Plan recognition is the problem of inferring the goals and plans of an agent from partial observations of her behavior (Cohen, Perrault, and Allen 1981; Pentney et al. 2006; Yang 2009). The problem arises in a number of applications, and has been addressed using a variety of methods, including specialized procedures (Kautz and Allen 1986; Avrahami-Zilberbrand and Kaminka 2005), parsing algorithms (Pynadath and Wellman 2002; Geib and Goldman 2009) and Bayesian networks inference procedures (Bui 2003). In almost all cases, the space of possible plans or activities to be recognized is assumed to be given by a suitable library of policies or plans.

Recently, two formulations have approached the plan recognition problem from a different perspective that replaces the need for a set of possible agent policies or plans, by an agent action model and a set of possible goals. The model expresses how the agent can go about achieving these goals and is used to interpret the observations. The result is a posterior probability distribution over the possible goals. In these approaches, the possible agent behaviors are encoded implicitly in the set of goals and action models, rather than explicitly as a library of plans. The advantage of these approaches to plan recognition is that they can leverage on model-based behavior generators; namely, planners. In (Ramirez and Geffner 2009; 2010), the model is classical planning model, namely, the initial state is fully known to agent and observer, and the actions have deterministic effects, while in (Baker, Saxe, and Tenenbaum 2009), the model is a Markov Decision Process (MDP), so that the states are fully observable, and actions have stochastic effects.

In this work, we extend the model-based approach to plan recognition over POMDP settings, where actions are stochastic and states are partially observable. The task is to infer a probability distribution over the possible goals of an agent whose behavior results from a POMDP (Partially Observable MDP) model. The model is shared between agent and observer except for the true goal of the agent that is hidden to the observer. The observations are action sequences that may contain gaps as some or even most of the actions done by the agent are not observed. We show that the posterior goal distribution can be computed from the value function over beliefs generated by a POMDP planner for each possible goal G. More precisely, executions are sampled from this value function, assuming that the agent tends to select the actions that look best, and the likelihood of the observations O given the goal G is approximated from these samples. In analogy to the other cases, the goal recognition problem over POMDPs is solved using an off-the-shelf POMDP planner. While POMDP planners do not scale up as well as MDP planners, and certainly much worse than classical planners, we show that still a rich variety of recognition problems involving incomplete information can be effectively modeled and solved in this manner. The expressive power and computational feasibility of the approach will be illustrated through a number of experiments over several domains.

The paper is organized as follows. We start with an example (Section 2), and then review previous approaches (Section 3), and POMDPs (Section 4). We then consider a preliminary formulation of POMDP goal recognition that assumes that all agent actions and observations are visible to

*Paper appearing also at Proc. IJCAI–11.

the observer (Section 5), and a more general form that assumes neither (Section 6). We then test the latter over several problems (Section 7) and summarize the contributions (Section 8).

Motivation

As an illustration of how a goal recognition problem can be naturally cast in the POMDP setting, consider an agent that is looking for an item A or B each of which can be in one of three drawers 1, 2, or 3, with probabilities $P(A@i)$ and $P(B@i)$ equal to:

$$P(A@1) = 0.6\ , P(A@2) = 0.4\ , P(A@3) = 0$$
$$P(B@1) = 0.1\ , P(B@2) = 0.6\ , P(B@3) = 0.3$$

The actions available to the agent are to open and close the drawers, to look for an item inside an open drawer, and to grab an item from a drawer if it's known to be there. Let us assume that the agent is a male, and hence that the probability that he doesn't observe the object in the drawer when the object is actually there is non-zero, say 0.2, but that the probability that he observes an object when it's not there is 0 indeed.

Let us assume that the possible goals G_1, G_2, and G_3 of the agent are to have item A, item B, or both, with priors 0.4, 0.4, and 0.2. We want to find out the goal posterior probabilities when the behavior of the agent is partially observed. In our setting, the observer gets to see some of the actions done by the agent, but not necessarily all of them. The observer must then fill up the gaps. Let us assume that it is observed that the agent opens drawer 1, then drawer 2, and then drawer 1 again; i.e.,

$$O = \{open(1), open(2), open(1)\}.$$

The most likely explanation of this observation trace is that the agent is looking for item A; else it wouldn't have started by looking in drawer 1 where the probability of finding B is 0.1. Then, it's likely that the agent didn't observe A in that drawer, that it closed it, and then looked for A in drawer 2. Then, probably the agent didn't find A in drawer 2, and thus looked again in drawer 1.

Indeed, the algorithm that we will describe, concludes that the posterior probabilities for the three possible goals are $P(G_1|O) = 0.6$, $P(G_2|O) = 0.1$, and $P(G_3|O) = 0.3$, with G_1 as the most likely goal.

Previous Approaches

As mentioned in the introduction, the problem of plan, goal, or activity recognition has been addressed in many ways, in most cases assuming that there is a library of possible plans or policies that represents the possible agent behaviors. The problem has been formulated in a variety of ways, as a deductive problem over a suitable logical theory (Kautz and Allen 1986), a matching problem over a suitable AND/OR graph (Avrahami-Zilberbrand and Kaminka 2005), a parsing problem over a grammar (Pynadath and Wellman 2002; Geib and Goldman 2009), and an inference task over a dynamic Bayesian network (Bui 2003).

Some recent approaches, however, attempt to map the *plan recognition problem* into *plan generation* to leverage on the performance of state-of-the-art planners. Ramirez and Geffner (2010) consider the problem of plan recognition over classical planning models where the goal of the agent is hidden to the observer. They show that the posterior distribution $P(G|O)$ over the possible agent goals G given a sequence of observations O, can be defined from the costs $c(G, O)$ of the plans that achieve G while complying with the observations O, and the costs $c(G, \overline{O})$ of the plans that achieve G while not complying with O. Indeed, they define the likelihood $P(O|G)$ as a monotonic (sigmoid) function of the difference in costs $c(G, \overline{O}) - c(G, O)$. Thus, when the best plans for G all comply with O, this difference will be positive, and the larger the difference, the larger the likelihood $P(O|G)$. In order to compute the posterior probabilities $P(G|O)$ for a set $\mathcal{G}$ of possible goals G, the likelihoods $P(O|G)$ are then derived in $2|\mathcal{G}|$ planner calls, and they are then plugged into Bayes rule along with the priors $P(G)$ to yield the posterior probabilities $P(G|O)$.

The other recent model-based approach to plan recognition is in the MDP setting where actions are assumed to have stochastic effects and states are fully observable (Baker, Saxe, and Tenenbaum 2009). Baker *et. al.* show that from the value function $V_G(s)$ that captures the expected cost from state s to the goal G, for every state s and goal G, it is possible to define the probability that the agent will take a given action a in s if her goal is G. From this probability $P(a|s, G)$ and simple manipulations involving basic probability laws, they derive the likelihood $P(O|s, G)$ that the agent performs a sequence of actions O given that she starts in s and pursues the goal G. As before, from the likelihoods and the goal priors $P(G)$, they derive the posterior probabilities $P(G|O)$ using Bayes rule. Once again, the main computational work is done by the planner, in this case an MDP planner, that must furnish the value function $V_G(s)$ for all goals G and states s. Notice that in both the classical and MDP formulations, *probabilities* are inferred from *costs*; in the first case, the costs $c(G, O)$ and $c(G, \overline{O})$, in the second, the expected costs $V_G(s)$. The formulation that we develop below takes elements from both formulations while extending them to the POMDP setting where actions are stochastic and states are just *partially observable*.

Background: Goal MDPs and POMDPs

Shortest-path MDPs provide a generalization of the state models traditionally used in heuristic search and planning in AI, accommodating stochastic actions and full state observability (Bertsekas 1995). They are given by

- a non-empty state space S,
- a non-empty set of goal states $S_G \subseteq S$,
- a set of actions A,
- probabilities $P_a(s'|s)$ for $a \in A$, $s, s' \in S$, and
- costs $c(a, s)$ for $a \in A$ and $s \in S$.

The goal states t are assumed to be absorbing and cost-free; meaning $P_a(t|t) = 1$ and $c(a, t) = 0$ for all $a \in A$. Goal MDPs are shortest-path MDPs with a known initial state s_0

and *positive action costs* $c(a, s)$ for all a and non-terminal states s. Shortest-path and Goal MDPs appear to be less expressive than *discounted reward* MDPs, where there is no goal, rewards can be positive, negative, or zero, and a parameter γ, $0 < \gamma < 1$, is used to discount future rewards. Yet, the opposite is true: discounted reward MDPs can be transformed into equivalent Goal MDPs, but the opposite transformation is not possible (Bertsekas 1995). The same holds for discounted reward POMDPs and Goal POMDPs (Bonet and Geffner 2009).

The solution to MDPs are functions π mapping states into actions. The expression $V^\pi(s)$ denotes the *expected cost* that results from following the policy π from the state s to a goal state, and it can be computed by solving a system of $|S|$ linear equations. The optimal policies are well-defined if the goal is reachable from every state, and corresponds to the policies π^* that minimize $V^\pi(s)$ over all states s. The optimal cost function $V^*(s) = V^\pi(s)$ for $\pi = \pi^*$, turns out to be the unique solution to the Bellman equation:

$$V(s) = \min_{a \in A} \left\{ c(a, s) + \sum_{s' \in S} P_a(s'|s) V(s') \right\} \quad (1)$$

for all $s \in S \setminus S_G$, and $V(s) = 0$ for $s \in S_G$. The Bellman equation can be solved by the *Value Iteration* (VI) method, where a value function V, initialized arbitrarily over non-goal states, is updated iteratively until convergence using the right-hand side of (1). The optimal policy π^* is the policy π_V that is *greedy* in the value function V

$$\pi_V(s) = \operatorname*{argmin}_{a \in A} \left\{ c(a, s) + \sum_{s' \in S} P_a(s'|s) V(s') \right\}. \quad (2)$$

when $V = V^*$. Recent variants of value iteration aim to exploit the use of lower bound (admissible) cost or heuristic functions to make the updates more focused and to achieve convergence on the states that are relevant only. One of the first such methods is *Real-Time Dynamic Programming* (RTDP), that in each trial simulates the greedy policy π_V, updating the value function V over the states that are visited (Barto, Bradtke, and Singh 1995). With a good initial lower bound V, RTDP and other recent heuristic search algorithms for MDPs, can deliver an optimal policy without even considering many of the states in the problem.

POMDPs (Partially Observable MDPs) generalize MDPs by modeling agents that have incomplete state information (Kaelbling, Littman, and Cassandra 1999) in the form of a prior belief b_0 that expresses a probability distribution over S, and a sensor model made up of a set of observation tokens Obs and probabilities $Q_a(o|s)$ of observing $o \in Obs$ upon entering state s after doing a. Formally, a *Goal POMDP* is a tuple given by:

- a non-empty state space S,
- an initial belief state b_0,
- a non-empty set of goal states $S_G \subseteq S$,
- a set of actions A,
- probabilities $P_a(s'|s)$ for $a \in A$, $s, s' \in S$,
- *positive* costs $c(a, s)$ for non-target states $s \in S$,
- a set of observations Obs, and
- probabilities $Q_a(o|s)$ for $a \in A, o \in Obs$, $s \in S$.

It is also assumed that goal states t are cost-free, absorbing, and fully observable; i.e., $c(a, t) = 0$, $P_a(t|t) = 1$, and $t \in Obs$, so that $Q_a(t|s)$ is 1 if $s = t$ and 0 otherwise. The *target beliefs* or goals are the beliefs b such that $b(s) = 0$ for $s \in S \setminus S_G$.

The most common way to solve POMDPs is by formulating them as completely observable MDPs over the *belief states* of the agent. Indeed, while the effects of actions on states cannot be predicted, the effects of actions on *belief states* can. More precisely, the belief b_a that results from doing action a in the belief b, and the belief b_a^o that results from observing o after doing a in b, are:

$$b_a(s) = \sum_{s' \in S} P_a(s|s') b(s'), \quad (3)$$

$$b_a(o) = \sum_{s \in S} Q_a(o|s) b_a(s), \quad (4)$$

$$b_a^o(s) = Q_a(o|s) b_a(s) / b_a(o) \quad \text{if } b_a(o) \neq 0. \quad (5)$$

As a result, the *partially observable* problem of going from an initial state to a goal state is transformed into the *completely observable* problem of going from one *initial belief state* into a *target belief state*. The Bellman equation for the resulting *belief* MDP is

$$V(b) = \min_{a \in A} \left\{ c(a, b) + \sum_{o \in Obs} b_a(o) V(b_a^o) \right\} \quad (6)$$

for non-target beliefs b and $V^*(b_t) = 0$ otherwise, where $c(a, b)$ is the expected cost $\sum_{s \in S} c(a, s) b(s)$.

Many of the methods used for solving POMDPs are MDP methods extended to deal with the infinite and dense set of possible belief states. In our experiments, we use RTDP-Bel (Bonet and Geffner 2000; 2009), which is a straightforward adaptation of RTDP to Goal POMDPs where states are replaced by belief states updated according to (6).

Goal Recognition: Complete Observations

Our first formulation of goal recognition over POMDPs is a direct generalization of the MDP account (Baker, Saxe, and Tenenbaum 2009). This account makes *two assumptions*. First, that the observation sequence $O = a_1, \ldots, a_n$ is *complete*, meaning that O contains *all the actions done by the agent up until* a_n, and hence, that there are no gaps in the sequence. Second, that *the states of the MDP are fully observable not only to the agent, but also to the observer.* The assumptions are pretty restrictive but serve to reduce the goal recognition problem to a simple probabilistic inference problem. In the POMDP setting, the second assumption translates into the (partial) observations gathered by the agent being visible also to the observer. Thus, in this setting, the observer gets two types of information: the *complete* sequence of actions O done by the agent, and the corresponding sequence of POMDP observation tokens $o \in Obs$ that the agent received. In the next section, we relax these two assumptions.

The POMDP is assumed to be known by both the agent and the observer, except for the actual goal G of the agent.

Instead, the set $\mathcal{G}$ of possible goals is given along with the priors $P(G)$. The posterior goal probabilities $P(G|O)$ can be obtained from Bayes rule:

$$P(G|O) = \alpha P(O|G)P(G) \tag{7}$$

where α is a normalizing constant that doesn't depend on G. The problem of inferring the posteriors $P(G|O)$ gets thus mapped into the problem of defining and computing the likelihoods $P(O|G)$. The key assumption is that if the agent is pursing goal G, the probability $P(a|b, G)$ that she will choose action a in the belief state b is given by the Boltzmann policy:

$$P(a|b, G) = \alpha' exp\{\beta\, Q_G(a, b)\} \tag{8}$$

where α' is a normalizing constant and β captures a 'soft rationality' assumption (Baker, Saxe, and Tenenbaum 2009): for large β, the agent acts greedily on Q_G (optimally if Q_G is optimal); for low β, the agent selects actions almost randomly.

The term $Q_G(a, b)$ expresses the expected cost to reach the goal G from b starting with the action a; i.e.,

$$Q_G(a, b) = c(a, b) + \sum_{o \in O} b_a(o) V_G(b^o_a) \tag{9}$$

where V_G is the value function for the POMDP assuming that the goal states are those in which G is true, $c(a, b)$ is the expected cost of action a in b, and $b_a(o)$ and b^o_a as defined above, stand for the probability that agent observes o after doing action a in b, and the probability distribution that results from doing a in b and actually observing o. The likelihood $P(O_i|b, G)$ of the observation sequence $O_i = a_i, \ldots, a_n$ given the belief b and the goal G, can be computed recursively as:

$$P(O_i|b, G) = \begin{cases} P(a_n|b, G) & \text{if } i = n, \text{ else} \\ P(a_i|b, G) \sum_o P(O_{i+1}|b^o_a, G)\, b_a(o)\,. \end{cases} \tag{10}$$

The likelihood $P(O|b, G)$ is then $P(O_i|b, G)$ for $i = 0$, which can be computed from the recursion and plugged into Bayes rule (7) to obtain the desired posterior goal probabilities $P(G|O)$. The POMDP planner enters into this formulation by providing the expected costs $V_G(b)$ to reach G from b, that are used via the factors $Q_G(a, b)$ for defining the probability that the agent will do the action a when in the belief state b (Eq. 8).

Goal Recognition: Incomplete Observations

In the account above, the information available to the observer contains both the sequence of actions O done by the agent, and the observations $o \in Obs$ that the agent receives from the environment. Moreover, the sequence of actions is assumed to be complete, so that all the agent actions are observed. In the account below, the sequence of actions O obtained by the observer may be incomplete and the observations received by the agent are not available.

As before, we assume a shared POMDP between agent and observer, except for agent goal G that belongs to the set $\mathcal{G}$ of possible goals but is hidden to the observer. Since the observation sequence $O = a_1, \ldots, a_n$ is not necessarily complete, we cannot assume that action a_{i+1} in O is the action that the agent did right after a_i. Yet, the posterior goal probabilities $P(G|O)$ can be derived using Bayes rule (7) from the priors $P(G)$ and the likelihoods $P(O|G)$ that can now be defined as

$$P(O|G) = \sum_\tau P(O|\tau) P(\tau|G) \tag{11}$$

where τ ranges over the possible executions of the agent given that she is pursuing goal G. Executions τ contain the complete sequence of agent actions.

We will say that *an execution τ complies with the observation sequence O* if the sequence O is embedded in the sequence τ. Defining then the probabilities $P(O|\tau)$ to 1 or 0 according to whether the execution τ complies with O or not, the sum in (11) can be approximated by *sampling* as

$$P(O|G) \approx m_0/m \tag{12}$$

where m is the total number of executions sampled for each goal G, and m_O is the number of such executions that comply with O.

For this approximation to work, executions τ for the goal G need to be sampled with probability $P(\tau|G)$. This can be accomplished by making the agent select the action a in a belief b with a probability $P(a|b, G)$ that results from the Boltzmann policy (8). As before, it is assumed that the POMDP planner, furnishes the value function $V_G(b)$ that encodes the expected cost from b to the goal.

Once the action a is sampled with probability $P(a|b, G)$, the resulting observation o, that is no longer assumed to be available to the observer, is sampled with probability $b_a(o)$. The resulting full traces $b_0, a_0, o_0, b_1, a_1, o_1, \ldots$ until the goal is reached are such that $b_{i+1} = b^o_a$ for $b = b_i$, $a = a_i$, and $o = o_i$, where a_i is sampled with probability $P(a_i|b_i, G)$, and o_i is sampled with probability $b_a(o_i)$ for $b = b_i$ and $a = a_i$.

The likelihoods $P(O|G)$ approximated through (11) are then plugged into Bayes rule (7) from which the posterior goal probabilities $P(G|O)$ are obtained.

The key computational burden in this account results from the calculation of the value function V_G over beliefs, that must be provided as before by the POMDP planner, and the simulated executions that need to be done for estimating the likelihoods $P(O|G)$ following (12).

Experiments

To evaluate the effectiveness of the goal recognizer described in Section 6, we used the POMDP solver GPT (Bonet and Geffner 2001) built around the RTDP-BEL algorithm. GPT supports a very expressive language to define POMDPs which have allowed us to test our approach over three challenging domains detailed next. We needed to make two modifications on the original GPT sources. We changed the code to have it built with the latest versions of the GNU C++ compiler and libraries, and added a method to simulate the policies computed by GPT. The software for the goal recognition part was implemented on PYTHON, making calls to GPT when necessary.

Name	$\|S\|$	$\|A\|$	$\|Obs\|$	$\|b_0\|$	$\|\mathcal{G}\|$	T
OFFICE	2,304	23	15	4	3	3.4
DRAWERS	3,072	16	16	6	3	4.5
KITCHEN	69,120	29	32	16	5	10.1

Table 1: Domains used in the evaluation. $|S|$ denotes number of states, $|A|$ number of actions, $|Obs|$ number of observations, $|b_0|$ cardinality of initial belief, $|\mathcal{G}|$ number of possible goals and T average time (in seconds) to compute $V_G(b_0)$ for each of the goals G.

Table 1 shows the number of states, actions, observations, goals, and possible initial states for each of the three domains we used to evaluate the proposed goal recognition scheme with incomplete observations. These are non–trivial POMDPs that feature uncertainty in the initial state, stochastic actions and stochastic sensors. We will describe the domains next.

The DRAWERS domain is the task described early in the paper. The agent goals are to be holding either an object named *A*, an object named *B* or both. The two objects are distributed in the drawers according to the probability distribution described in Section 2. The agent can open drawers, look for a particular object inside an open drawer and pick an object. All actions have unitary cost. There is a small chance – 30 out of every 100 times – that she does not find the object she is looking for, even if the object actually is in the inspected drawer. Singleton goals are considered to have equal prior probability (0.4). For the joint goal we set its prior probability to the product of the singleton goals. As an illustration, if the agent is observed to "look into drawer #3" then it is more likely the goal "hold B", since A can't be initially placed in drawer #3.

In the OFFICE domain, adapted from (Bui 2003), the agent being observed is a researcher who arrives early to her lab, which consists of two rooms: one is her office, where she has her workstation and a cabinet to store her coffee cup and blank paper. The other is the club, where the lab coffee machine and printer are placed. The two rooms are connected by a corridor. The researcher is initially on the corridor. The printer can start either out of paper, clogged, both or none. The agent goals are either to print an article, have a cup of coffee or both. To print an article, the researcher needs to get to her workstation, send the file to the printer queue and get to the printer to retrieve it. There is a small chance – 20 out of every 100 times – that the printer will get clogged while trying to print the article. If the printer is out of paper, the file is kept on the printer queue. In this case, the agent will need to fetch blank paper from the cabinet in her office. When the printer is clogged, the agent will have to execute several actions to service it. To have coffee, the agent needs to get the cup from the cabinet in her office and then walk to the coffee machine. As in the DRAWERS domain, all actions have a cost of 1, singleton goals have equal prior probability, and the joint goal prior probability is the product. For example, if the agent is observed to execute the actions "walk to workstation, walk from corridor into club" then the most likely goal will be "print article", since the agent doesn't need to get to the workstation if she's pursuing the goal "have coffee".

In the KITCHEN domain the agent is trying to cook one out of five possible dishes. There are ingredients $i1$, $i2$, $i3$, $i4$ which are placed at random on two cupboards. Each dish requires up to three different ingredients which are required to be mixed in a bowl. The agent can inspect the cupboards the find the ingredients it needs, having to move first in front of the cupboard of interest. Additionally the agent needs to get hold of three different objects – a tray, a pan and a pot – which are all located in a third cupboard. Whenever a recipe involves boiling or frying an ingredient, or a mix of them, the agent needs to place the required objects on the stove. The agent can navigate freely between the locations of the bowl, the stove or the cupboards and can carry as many ingredients as she sees fit. All goals have the same prior probability and action costs are uniform. Thus if the agent is observed to "take pan, take $i2$" then the most likely goals will be those dishes which require to fry a mix of ingredients that includes $i2$.

The synthetic dataset was built as follows. For each domain and goal, we first computed the value function V_G for each possible goal G using GPT. Then, we sampled 100 executions of the greedy policy based on V_G for each possible goal $G \in \mathcal{G}$. From these 100 sampled executions, 10 observation sequences O were obtained by sampling randomly 30%, 50% or 70% of the actions over an also randomly selected execution.

In the table and figures, we don't report the posterior goal distributions $P(G|O)$, but just the resulting *binary classifier* that maps the observation sequences O into sets of *most likely* goals G. These are the goals G that maximize the posterior probability $P(G|O)$. We denote this binary goal recognizer as GR(m, β), where m and β stand for the two parameters of the algorithm: the number of samples for each goal used in the approximation of $P(O|G)$, and the β coefficient that expresses the level of noise in the action selection. The set of most likely goals are those G' that verify $P(G') = \max_{G \in \mathcal{G}} P(G|O)P(G)$ [1].

The classification instances are the pairs $\langle O, G \rangle$ over all the observation sequences O and goals G. An instance is positive (P) if O was generated with G, and negative (N) otherwise. A true positive (TP) is a positive instance classified as positive, while a false negative, is a negative instance classified as positive. True negatives (TN) and false negatives (FN) are defined in a similar manner. The numbers of instances in these different classes provide the standard measures for evaluating the quality of a classifier. In particular, TPR, FPR, ACC, and PPV measure the True Positive Rate, False Positive Rate, Accuracy, and Precision of a classifier, defined as TP/P, FP/N, $TP + TN/P + N$, and $TP/TP + FP$ respectively.

Figure 1 shows the aggregate results of the goal recognizer GR(m, β) over all domains, in the form of a ROC graph (Fawcett 2006) that plots TPR vs. FPR. As it can be seen, the performance of the goal classifier approaches the

[1] We consider two real numbers x, x' to be equal whenever $|x - x'| < \epsilon$, where ϵ is set to 10^{-7}

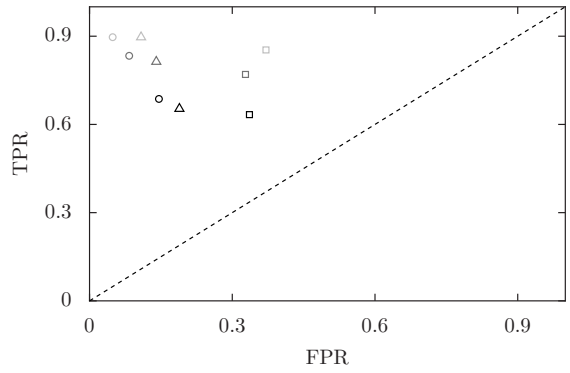


Figure 1: ROC graph showing the resulting goal classifier $\text{GR}(m, \beta)$ for different m and β values (number of samples and noise level in action selection). Squares, triangles and circles denote different m values: 100, 1000 and 10000. Black, gray, and light gray denote different β values: 1, 10, 40. Results for the random guessing strategy are represented by the dotted line.

Domain	Obs %	L	T	ACC	PPV	TPR
office	30	4.9	24.6	0.99	0.97	1.00
	50	7.6	24.7	1.00	1.00	1.00
	70	10.8	24.8	1.00	1.00	1.00
kitchen	30	3.8	95.2	0.86	0.73	0.73
	50	5.8	95.1	0.93	0.85	0.85
	70	8.3	95.2	0.98	0.95	0.95
drawers	30	2.9	38.8	0.84	0.77	0.77
	50	3.9	38.8	0.87	0.80	0.80
	70	6.0	38.8	0.96	0.93	0.93

Table 2: Performance of $\text{GR}(m = 10,000, \beta = 40)$. For each domain and observation level we report the average length of observation sequences O (L), the average time in seconds to process one observation sequence (T) and the average accuracy (ACC), precision (PPV) and True Positive rate (TPR).

optimal vertex $(1, 0)$ as the number of samples m becomes large (see caption for details). Performance is very good – high TPR, low FPR – for $m \geq 1000$ and high β values. However we can also see that even with a substantial amount of samples and the Boltzmann policy being almost greedy – $\beta = 40$ – we cannot lower FPR nor raise TPR.

Table 2 offers a detailed picture of the performance of the goal recognizer for values $m = 10,000$ and $\beta = 40$. In all domains we see how the accuracy of goal recognition increases as more information is conveyed by the input observation sequence. It is remarkable that $\text{GR}(m = 10,000, \beta = 40)$ achieves almost perfect recognition on the OFFICE domain, having some trouble with the shortest and sparsest observation sequences.

Runtime is determined by the number of possible goals $|\mathcal{G}|$, the number of samples taken m and the value for β. Processing one observation sequence involves simulating – the time required to compute $V_G(b)$ is reported on Table 1 – the Boltzmann policy m times $|\mathcal{G}|$, hence the similarity between the run-times for OFFICE and DRAWERS, which have the same number of possible goals. Runtime also increases as m increases and as β *decreases*. While the reasons for the former are quite obvious – the more simulations to check whether they are compatible with O the more computation – the former can seem a bit surprising. Runtime grows as β decreases since action selection becomes noisier, so the execution trace gets longer.

While the goal recognizing accuracy is very good, it can't be perfect since there may be observation sequences O which result ambiguous. For example, in the OFFICE domain, the goal recognizer assigns equal $P(O|G)$ to all goals, when confronted with the sequence "*walk from corridor to lab*, *walk from lab to corridor*, *walk from corridor to club*". While the joint goal is discarded because of its lower $P(G)$, there is no other information available that supports rejection of either of the singleton goals "read article" and "have coffee".

Extensions

The model above for goal recognition over POMDPs is simple but expressive, yet there are a number of natural extensions, some of which we describe next.

• *Agent POMDP model partially known by observer:* in the above formulation, the agent POMDP model is known by the observer except for the hidden goal. Incomplete information about the *initial belief state* b_0 of the agent, however, can be accommodated as well. The simplest approach is to define a set B_0 of possible initial belief states b_0 each with a probability $P(b_0)$. The formulation can then be generalized to deal with both hidden goals G and hidden initial belief states b_0, and the posterior probabilities over the collection of such pairs can be computed in a similar manner.

• *Failure to observe and actions that must be observed:* as argued in (Geib and Goldman 2009), information about actions a that if done, must always be observed, is valuable, as the absence of such actions from O, imply that their were not done. This information can be used in a direct manner in the formulation above by just adjusting the notion of when a sample execution τ complies with O. In the presence of *must-see* actions, executions τ comply with O when τ embeds O, *and* every *must-see* action appears as many times in τ as in O.

• *Observing what the agent observes:* we have assumed that the observer gets a partial trace of the actions done by the agent and nothing else. Yet, if the observer gets to see some of the observation tokens $o \in Obs$ gathered by the agent, she can use this information as well. In particular, the number m_O in (12) would then be set to the number of sampled executions for G that comply with both O and Obs.

• *Noise in the agent-observer channel:* if the observer gets to see the actions done by the agent through a noisy channel where actions can be mixed up, the problem of determining where a sample execution τ complies with the observations O is no longer a *boolean* problem where $P(O|\tau)$ is either 0 or 1, but a probabilistic inference problem that can be solved in linear-time with Hidden Markov Model (HMM) algorithms, that would yield a probability $P(O|\tau)$ in the interval $[0, 1]$. For this, the model must be extended with probabilities $P_O(o|a)$ of observing token o from the execution of

action a, and hidden chain variables $t_i = j$ expressing that the observation token o_i in $O = o_1, \ldots, o_n$ has been generated by action a_j in the sample execution $\tau = a_1, \ldots, a_m$.

Discussion

We have formulated and tested a new formulation of goal recognition for settings where the observed agent can be modeled as acting using a POMDP whose goal is hidden to the observer. The posterior goal probabilities G for the hidden goals $G \in \mathcal{G}$ are computed from Bayes rule using the priors $P(G)$ and likelihoods $P(O|G)$ that are approximated in two steps: using first a POMDP planner to produce the expected costs V_G from beliefs to goals, and using these costs to sample the possible executions for each goal G. A number of direct extensions have also been discussed, like the integration of uncertainty about the initial belief of the agent, noise in the agent-observer channel, and observations obtained by both the agent and observer. A number of experiments have been reported and the results appear promising. The software and the domains will be made available.

References

Avrahami-Zilberbrand, D., and Kaminka, G. A. 2005. Fast and complete symbolic plan recognition. In *Proceedings of IJCAI*, 653–658.

Baker, C. L.; Saxe, R.; and Tenenbaum, J. B. 2009. Action understanding as inverse planning. *Cognition* 113(3):329–349.

Barto, A.; Bradtke, S.; and Singh, S. 1995. Learning to act using real-time dynamic programming. *Artificial Intelligence* 72:81–138.

Bertsekas, D. 1995. *Dynamic Programming and Optimal Control, Vols 1 and 2*. Athena Scientific.

Bonet, B., and Geffner, H. 2000. Planning with incomplete information as heuristic search in belief space. In *Proc. of AIPS-2000*, 52–61. AAAI Press.

Bonet, B., and Geffner, H. 2001. Gpt: A tool for planning with uncertainty and partial information. In *Proc. IJCAI Workshop on Planning with Uncertainty and Partial Information*.

Bonet, B., and Geffner, H. 2009. Solving POMDPs: RTDP-Bel vs. Point-based algorithms. In *Proceedings IJCAI-09*, 1641–1646.

Bui, H. 2003. A general model for online probabilistic plan recognition. In *Proc. IJCAI-03*, 1309–1318.

Cohen, P. R.; Perrault, C. R.; and Allen, J. F. 1981. Beyond question answering. In Lehnert, W., and Ringle, M., eds., *Strategies for Natural Language Processing*. LEA.

Fawcett, T. 2006. An introduction to ROC analysis. *Pattern Recognition Letters* (27):861–874.

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

Kaelbling, L. P.; Littman, M.; and Cassandra, A. R. 1999. Planning and acting in partially observable stochastic domains. *Artificial Intelligence* 101:99–134.

Kautz, H., and Allen, J. F. 1986. Generalized plan recognition. In *Proc. AAAI-86*, 32–37.

Pentney, W.; Popescu, A.; Wang, S.; Kautz, H.; and Philipose, M. 2006. Sensor-based understanding of daily life via large-scale use of common sense. In *Proc. AAAI-06*.

Pynadath, D., and Wellman, M. 2002. Generalized queries on probabilistic context-free grammars. *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 20(1):65–77.

Ramirez, M., and Geffner, H. 2009. Plan recognition as planning. In *Proc. 21st Intl. Joint Conf. on Artificial Intelligence*, 1778–1783. AAAI Press.

Ramirez, M., and Geffner, H. 2010. Probabilistic plan recognition using off-the-shelf classical planners. In *Proc. AAAI-10*. AAAI Press.

Yang, Q. 2009. Activity Recognition: Linking low-level sensors to high-level intelligence. In *Proc. IJCAI-09*, 20–26.

Towards a System Architecture for Recognizing Domestic Activity by Leveraging a Naturalistic Human Activity Model

Michele Dominici
INRIA, Centre Inria Rennes - Bretagne Atlantique
Campus Universitaire de Beaulieu
35042 Rennes Cedex, France
Michele.Dominici@inria.fr

Myriam Fréjus and **Julien Guibourdenche**
EDF Recherche & Développement
1 Avenue du Général de Gaulle
92141 CLAMART Cedex, France
firstname.lastname@edf.fr

Bastien Pietropaoli
INRIA, Centre Inria Rennes - Bretagne Atlantique
Campus Universitaire de Beaulieu
35042 Rennes Cedex, France
Bastien.Pietropaoli@inria.fr

Frédéric Weis
Université de Rennes 1, IRISA
Campus Universitaire de Beaulieu
35042 Rennes Cedex, France
Frederic.Weis@irisa.fr

Abstract

Existing activity recognition approaches in the smart home domain suffer from poor human activity models. Combining expertise from cognitive ergonomics and ubiquitous computing, we discuss the hard technical challenges to address when leveraging a realistic model of human activity. We present the architecture of a prototype smart home system that we are developing and show the gap that exists between our current capabilities in terms of contextual-knowledge extraction and the complexity of the targeted activity recognition. To fill this gap, we propose and discuss the integration of PHATT, an existing algorithm for plan recognition, into our system in order to mine additional information from the dynamics of context.

1 Introduction and Motivation

A *smart home* is a residence equipped with information-and-communication-technology devices conceived to collaborate in order to anticipate and respond to the needs of the occupants, working to promote their comfort, convenience, security and entertainment while preserving their natural interaction with the environment (Aldrich 2003).

When talking about natural interaction, one of the most precious resources to preserve is user attention: during their activities, users should be supported *invisibly*, reducing interruptions and explicit interactions with the system as much as possible. In order to achieve these goals, smart home systems must be able to take into account the *context*, that is the implicit situational information that influences human behavior (Roy et al. 2010), recognizing people activities to provide adapted functionalities. For example, under some conditions, knowing that an inhabitant is executing some long-lasting static activity in a room can suggest that the system should turn on the room's heating and turn off the other rooms' lights.

Existing solutions for human activity recognition often rely on data coming from wearable sensors or video cameras (Chen and Nugent 2009), technologies that are difficult to deploy and get accepted in real-world households. Furthermore, these solutions address the problem of activity pattern discovery directly on raw sensor data or video streams, exploiting data mining techniques to extract recurring patterns in the raw data and to predict or classify future observations, as explained in (Kim, Helal, and Cook 2010). The resulting systems fail to provide adapted services to people in real-world scenarios, as the "gap" between the captured context and the complexity of human behavior is too large. We believe that the main reasons are the poverty (or absence) of the underlying models of human behavior and activities, which don't handle some fundamental aspects of the reality, and/or the lack of computing models taking advantage of these aspects.

To address these issues, we started an interdisciplinary project that brings together researchers from the fields of ubiquitous computing and cognitive ergonomics. Our aim is to develop a smart home system that is able to prevent energy waste and preserve inhabitants' comfort, leveraging on realistic human activity models. Our hypothesis is that human activity models have to be taken into account as challenging implication for informatics, although they shall not be directly integrated into computing models.

In this paper, our contribution is threefold. In Section 2, we present some of the challenging dimensions of human activity, which are not handled by most existing approaches, emerging when considering activity as a situated process, relying on actualization of concerns, and integrated in a network of interactions. In Section 3, we present a functional architecture that is designed to extract high-level situations from low-level raw sensor data. We show the need for an additional activity recognition mechanism and a system architecture that leverages the ubiquitous computing principles and that is at the core of the prototype system that we are developing. In Section 4, we propose to adapt PHATT, an existing algorithm for plan recognition (Goldman, Geib, and

Miller 1999; Geib and Goldman 2009), to be integrated into our architecture, in order to start addressing the challenge of providing adapted functionalities, which are well suited to the complexity of domestic activity. Section 5 illustrates the issues that remain unsolved and that may benefit from exchanging with the GAPRec and ICAPS research communities, while Section 6 concludes the paper.

2 Human Models of Domestic Activity

Recent naturalistic studies (Baillie and Benyon 2008; Crabtree and Rodden 2004; Guibourdenche et al. 2011; Poizat, Fréjus, and Haradji 2009; Salembier et al. 2009) provided some fundamental knowledge for a deeper empirical understanding of human domestic activity. Those studies aimed at orienting the design of ambient systems on the basis of real activity models and definitions of activity contexts, from the inhabitants' points of view. These formal descriptions of real activities and people's contexts are prerequisite for building appropriate applications (Greenberg 2001). These models also raise different issues challenging technical models for activity recognition.

Many existing technical models for activity recognition consider human activities as sequences of targeted actions that are always executed in the same order and which are never concurrently executed or interleaved with actions corresponding to other activities (Gu et al. 2009). Instead, we conducted our work in reference to the course of action empirical research program (Theureau 2003). This theoretical framework, as well as naturalistic studies, demonstrates that (domestic) activity is opportunistic. Inhabitants frequently interrupt a particular task for a while in order to accomplish another one. Individual activity at home is constituted of multiple lines of different concerns which structure a kind of fuzzy involvement in the activity. For example, a mother can be ironing while following a TV-show and looking after children playing at the first floor. Inhabitants manage several activities at the same time with several underlying concerns, which take part in their individual context. Activity is never built according to a pre-established and hierarchical plan but is constantly reoriented according to inter-individual interactions and interactions with the physical environment. This raises design issues relative to the gap between this complex human context and the context of the system based upon an environmental capture.

In addition, the same behavior (e.g. closing shutters) can have several meanings (e.g. reducing the brightness in a room, ensuring some privacy, increasing the sense of safety, reducing the temperature inside the house). This slight gap is due to the asymmetrical relation between environment as raw material, and situation as experienced environment through the individual's activity. Thus, the model has to integrate several layers of inference from a low level (e.g., a shutter is being closed) to a high level (e.g., Julie wants to have more cosiness), the latter the more problematic. Some situations can cause the system to an inability to determine the appropriate action to take; thus, designing a context-aware system implies designing the interaction with the user in order to manage uncertainty (explicit interaction, validation, etc).

Another design limitation rises from the impossibility for a smart home system to act according to deterministic rules only (either manually provided or automatically extracted through machine learning techniques). In our precedent example, a rule such as "closing shutters when night falls" can't be adapted to the several meanings underlying the closing of shutters. However, some solutions are based upon such design principles (Gu et al. 2004; Campo et al. 2006; Bonhomme et al. 2008b; 2008a), but they encounter difficulties to be adapted to inhabitants' practices. Even though some recurrent activities (e.g., cooking, taking children to bed, watching night shows on TV) can be observed at day or week scales, they nevertheless seem to be always accomplished differently, at different times or in a different order. Routines illustrate the recurrence of concerns, not the execution of schemes of action, as some works assume (Chan et al. 2008).

Furthermore, individual and collective scales of activity are intertwined, *mutually* (Crabtree and Rodden 2004; Poizat, Fréjus, and Haradji 2009) and *conflictually* (Baillie and Benyon 2008) giving shape to one another. For example the cleaning can be initiated by an individual and finished by another, the latter doing a different way than what the former had previously thought. Therefore, the system design can't rely on a human activity considered only as individual, a lonely man doing one thing at a time.

Moreover, the activity can't be strictly associated with a specific space (Guibourdenche et al. 2011): families are distributed across multiple scales of physical spaces (floors, rooms, systems of tools, voices, noises). During a local activity (for example, a mother doing the ironing and watching TV in the living room), concerns may refer to different places or people (as supervising children in the example: the mother is also concerned with her daughter alone upstairs). Those characteristics imply that the system can't consider only a local point of view on the activity and must integrate local and global points of views.

Now our aim is to specify an architecture capable of integrating these various constraints.

3 Architecture

In this Section, we present the architecture of a prototype system that we are developing. The aim of our system is to capture physical information from the environment, extract higher-level concepts and combine them to infer human situations and activities, with the ultimate goal of semi-automatically managing household appliances and provide additional functionalities that allow saving energy while preserving comfort. We first present the system architecture, which relies on the principles of the ubiquitous computing paradigm (Weiser 1993) and draws its inspiration from the four-layer model described in (Coutaz et al. 2005), and then highlight the need for an additional activity recognition mechanism.

3.1 Layered architecture

To achieve the goals of our scenario, we decided to adopt the human-computer interaction paradigm called *ubiquitous*

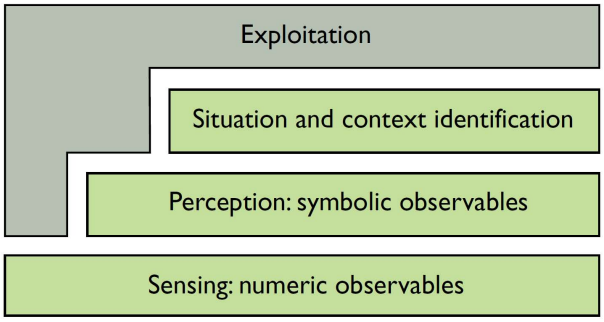


Figure 1: *The four-layer model for context-aware applications proposed by (Coutaz et al. 2005).*

computing. The aim of this paradigm is to seamlessly and invisibly integrate in the physical environment a multitude of digital devices that provide services to the users without asking for their attention (Weiser 1993). To this end, ubiquitous computing applications are typically context-aware, where the word context is used to address any static or dynamic condition that concerns the digital, physical and user-related environments in which a context-aware application is executed. In (Coutaz et al. 2005), a four-layer model is suggested to build context-aware applications, as showed in Fig. 1. The first layer, *sensing*, corresponds to the raw data sensed from the environment. The second layer, called *perception*, can be interpreted as an abstraction of the raw data. *Situation and context identification*, the third layer, concerns the context itself and the situations occurring in the home. The top layer, called *exploitation*, provides contextual information to applications. Our work is partly based on this model.

Considering the aforementioned model, the first layer of our system should be simply composed of sensors, but some constraints have to be fitted. In order to reduce the global system cost and to protect the inhabitants' privacy, the number of sensors dispatched in the environment has to be reduced as much as possible. However, a huge number of different sensors are required to sense context pieces and redundancy can significantly increase the reliability of the sources. With this idea in mind, the sensors will be grouped in nodes, as showed in Fig. 2. These nodes are able to preprocess the data with simple computation such as minimum, maximum and average. They also enable the sensors to communicate, using, for instance, 6LowPAN (IPv6 over LoW Power wireless Area Networks), which is specifically designed for embedded systems (Shelby and Bormann 2009). Another benefit of using nodes is the optimization of energy consumption due to radio communications. This is not negligible as most of the nodes will be running on batteries.

In the second layer of Coutaz' model, the raw data are processed to obtain more abstract data about context and occurring situations. The aggregation of raw data is realized thanks to a data fusion algorithm. The data fusion algorithm that we adopted is called the *belief functions theory* or *theory of evidence*. More specifically, the transferable belief model from (Smets and Kruse 1996) is used to aggregate data from

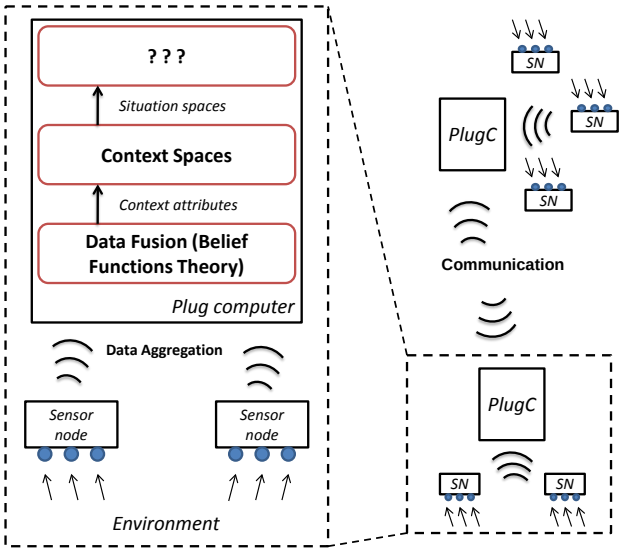


Figure 2: *The system architecture – The sensor nodes send aggregated data to the plug computers, which are in charge of performing sensor data fusion, to produce the context attributes, and context spaces reasoning, to infer the ongoing situation spaces. An additional processing step is needed to perform activity recognition.*

homogeneous as well as heterogeneous sources. The aim in this layer is to extract from raw sensing pieces of higher-level contextual information. Some existing solutions also adopt a similar approach. For instance, in (Ricquebourg et al. 2007), a pressure sensor on a chair, an omnidirectional webcam and a tracking sensor are used to determine the posture of a person. Another example is given in (Chahuara, Vacher, and Portet 2010) with the localization of inhabitants using microphones, presence sensors and contact sensors on house furnishing doors.

The bridge between the second and the third layer is realized integrating the results of sensor data fusion into a context model called *Context Spaces*. This model uses geometrical metaphors to describe context and situations, relying on the following concepts (Padovitz, Zaslavsky, and Loke 2006): the *context attributes*, the *application space*, the *situation spaces* and the *context state*. The *context attributes* are information types that are relevant and obtainable by the system; in our case, the context attribute values are provided by the perception layer, together with a degree of *confidence* on them, needed to cope with the intrinsic uncertainty of sensing systems in real world scenarios. The *application space* is a multi-dimensional space made up of a domain of values for each context attribute. The *situation spaces* are subspaces of the application space defined over regions of acceptable values of selected context attributes; situation spaces model real-life situations, e.g., "the whole family is in the kitchen" or "a person is ironing". A *context state* is the collection of current context attribute values at a given moment (Padovitz, Zaslavsky, and Loke 2006).

In the *situation and context identification* layer, the con-

text state provided by the perception layer is analyzed to infer the ongoing situation spaces (representing real-life situations) and also produce a measure of confidence in their occurrence. As the same context state can correspond to several different situation spaces (and vice versa), reasoning techniques are needed to discern the actual ongoing real-life situations in spite of uncertainty (Padovitz, Zaslavsky, and Loke 2006; Padovitz 2006).

Unfortunately, the computations required by the second and the third layers to obtain abstract data and to analyze context and situations are too heavy for our nodes to be processed on. To remedy to this problem, more powerful nodes acting like sinks are used. These nodes are small "plug and play" computers called *plug computers* (ref. Fig. 2). Their role is to gather data from sensor nodes and to perform data fusion, required to produce the context attributes, and context space reasoning, used to identify ongoing situations.

As explained in (Coutaz et al. 2005), the *exploitation* layer acts as an adapter, allowing applications to address to the infrastructure their requests for context services at a high level of abstraction. In our architecture, this layer will provide information about context to augmented appliances, which will adapt their behavior in a semi-automatic way.

3.2 Need for an additional layer

In our smart home distributed system, applications will directly execute on physical objects and household appliances, adapting their behavior to save energy and preserve inhabitant comfort. To this end, they will exploit context services, provided by the underlying infrastructure, to gain knowledge about the context and to provide the inhabitants with relevant information in a suitable way, following the ubiquitous computing principles (Weiser 1993). In our current implementation, the highest-level contextual information is produced by the situation and context identification layer exploiting the context spaces theory: the result is a set of occurring situations. As we explained in Sect. 2, the same situation can correspond to several different human activities and the same activity can require different forms of assistance depending on the particular situation. Thus, the artifact of situation space currently provided by our context spaces reasoning may not be sufficient to provide the targeted kind of assistance to inhabitants. To fill the gap between the situation spaces and the higher-level contextual information that we target, we need an additional mechanism that extracts a higher level of contextual knowledge from the underlying layer. Since the context spaces reasoning mechanisms only exploit the static contextual information provided by the perception layer, we need to mine additional information from the dynamics of the context. The main idea is that the context spaces theory provides very powerful modeling and reasoning mechanisms, but it can hardly handle the dynamism of context. Some techniques for context verification are developed in the theory, which help solve some ambiguities leveraging on historical context, namely, exploiting the *situation natural flow* (Padovitz et al. 2007). Furthermore, an extension to the context spaces theory has been proposed to perform context prediction (Boytsov, Zaslavsky, and Synnes 2009). Even though these techniques suggest the promising idea that context can be iteratively refined to solve ambiguities, they rely on the assumption that the real-world context obeys the same laws of a point following a trajectory inside a space. This assumption is too restrictive when willing to model the context dealing with the complex behavior of the inhabitants of a house, which often presents quite unpredictable evolutions. A different mechanism has to be adopted, which is able to capture relevant information from the context dynamics and to provide likely explanations for the observed situation sequences. The next Section presents an existing plan recognition algorithm and a way to adapt it to be integrated into our system in order to achieve these goals.

4 Activity Recognition using PHATT

In this section, we present PHATT, an algorithm introduced by Goldman, Geib and Miller in (Goldman, Geib, and Miller 1999) to perform plan recognition, and its application to our architecture. In order to do this, we first present the hierarchical task network planning problem, which is "inverted" by PHATT to perform plan recognition. Then, we show how PHATT can be adapted to be integrated into our system, in order to capture relevant information from the context dynamics.

4.1 Hierarchical Task Network Planning (HTN)

A *Hierarchical Task Network (HTN) planning problem* consists in automatically generating a *plan* starting from a set of *tasks* to execute and some constraints (La Placa, Pigot, and Kabanza 2009; Ghallab, Nau, and Traverso 2004). The problem relies on the specification of a plan library made of two components: the *tasks* to execute, which can be *primitive* if they don't ask for any further planning or *open*, otherwise, and the *methods*, which are prescriptions of how decomposing a task in (partially-) ordered sub-tasks. Note that a same task can be decomposed using different methods, thus resulting in different sub-task sequences. HTN planning proceeds by decomposing non-primitive tasks recursively into smaller and smaller subtasks, until primitive tasks are reached that can be performed directly.

4.2 PHATT

(Goldman, Geib, and Miller 1999; Geib and Goldman 2001; 2009) present PHATT, an algorithm for plan recognition based on a model of plan execution. The principle behind the algorithm is to perform plan recognition relying on three phases: defining the plan library, modeling the plan execution and recognizing the current execution, starting from the observations. The plan library is modeled like in the HTN planning problem presented above. The plan execution is modeled as a stochastic, generative model that selects actions to perform from a set of enabled primitive tasks called *pending set*, which is dynamically defined depending on the previous actions performed by the agent, the agent's goals and the plan library (Geib and Goldman 2009). Assuming this model of plan execution, PHATT takes as input a sequence of observations, which correspond to agent's actions, and generates the set of all possible explanations for the

observed sequence of primitive tasks, in terms of executed plans and, thus, goals. It then uses Bayesian inference to calculate the probabilities of the generated explanations and goals.

4.3 Integration of PHATT with the Existing Architecture

In Sect. 3, we saw that the situation and context identification layer of our architecture, implemented exploiting the context spaces theory, lacks effective modeling and recognition of the temporal dimension of the activities. In this Section, we described an algorithm for plan recognition called PHATT; we propose now to adopt that algorithm to provide a way to model complex activities that develop over time. To this end, we show the advantages and a way of adapting PHATT to be integrated into our existing architecture, in order to start addressing the challenge of modeling and recognizing complex activities. We also provide an example that better explains the proposed modifications to the algorithm and the overall approach.

Critical aspects of PHATT and adaptation As showed in Sect. 3, the highest level of abstraction that our existing architecture provides is given by the situation spaces, whose abstraction is realized by the context spaces reasoning mechanism, which provides the set of ongoing high-level situations together with a value of confidence in their actual occurrence. As we said, we propose to integrate PHATT into our system in order to recognize complex activities that develop over time. Existing activity recognition approaches using *hidden Markov models* or *conditional random fields* consider the primitive tasks as elementary actions that can be performed by a person, e.g., use a spoon or a knife (Kim, Helal, and Cook 2010). Even in some of the proposed applications of PHATT, the primitive tasks model elementary actions that can be either directly observed or inferred by their effects (Goldman, Geib, and Miller 1999; Geib and Goldman 2001; 2009). In our case, the existing architecture already includes three layers that abstract high-level situation spaces from the elementary "observations" provided by sensors. For this reason, we propose to replace the primitive tasks of the plan library used by PHATT, which represent the actions that can be observed, with the situation spaces of the context spaces model. In this way, the observations are not elementary actions like those proposed in the original PHATT description (Goldman, Geib, and Miller 1999; Geib and Goldman 2001; 2009), but high-level situations occurring in the smart home. That is, situation spaces become the partially-ordered steps in which methods and root-level taks are decomposed.

Advantages of PHATT We now show the aspects of the complex activity that can be modeled and handled by PHATT and its underlying model of plan execution. In Sect. 2, we presented the domestic activity as opportunistic and constituted of multiple lines of different concerns. This implies that a same person can be preoccupied by multiple concerns at the same time. We also said that activity is never built according to a pre-established and hierarchical plan but is constantly reoriented according to inter-individual interactions and interactions with the physical environment. These aspects may look as being in contrast with the kind of plan library adopted by PHATT, which assumes that the activity can be modeled and fully specified as a hierarchy of alternative partially ordered sequential actions. Instead, even though PHATT relies on this kind of plan library, it adopts a separate model for plan execution, as showed in the previous Section. This model considers the possibility that several root-level tasks are executed in an interleaved fashion and it is thus able to produce explanations that contain multiple high-level goals.

In Sect. 2, we also said that the same behavior of a person can have several motivations, thus making it important to integrate several layers of inference from a low to a high contextual level. PHATT provides an additional contextual inference mechanism to our existing architecture, allowing to logically abducting the possible goals of inhabitants starting from their dynamic behavior. Starting from this knowledge about a person's goal, it may be easier to make decisions about the kind of assistance to provide in a particular situation, since the knowledge about the ongoing situations is in some ways enriched with their "motivation".

Example scenario We presented the advantages of PHATT and a way of integrating it into our existing architecture. We now show an example scenario, illustrating how to model it and recognize the ongoing activities using the previously presented tools. Notice that the scenario we chose is simple enough to allow an effective modeling and recognition of the involved activities using PHATT, while still being enough difficult to handle to represent an improvement to our existing architecture's capabilities (not exploiting PHATT). This scenario does not pretend to cover all the challenging aspects of the activity, which we presented in Sect. 2. Further work will investigate the aspects that we do not consider in this scenario. In Sect. 5, we provide some prospects about possible further adaptations of PHATT in order to model and recognize more complex activities.

Suppose John is involved in the concern of doing the housework. For this, he puts his house in order and does the washing. The washing machine is in a separate room that can be reached walking through a corridor, which also leads to other different rooms. John collects the washing from the whole house, reaches the laundry room with his arms loaded, and then turns the light on in the room. He loads the washing machine, turns it on, and then walks away to continue the housework, turning the light off. After some time, he decides to come back and check whether the washing cycle is over, discovering that it is not. This time, he leaves the light on, since he knows that he'll soon come back to unload the machine, as few minutes of washing are left. When he goes back to the kitchen, he notices that it is time to cook the lunch, so he opens the fridge to check what food is in it and then turns on his laptop to look for a recipe. In the meanwhile, the washing machine cycle is over and the light in the laundry room is still on. He finally decides to go back to the room and unload the machine. Then, he leaves the room, carrying the clean washing, and turns the light off, having

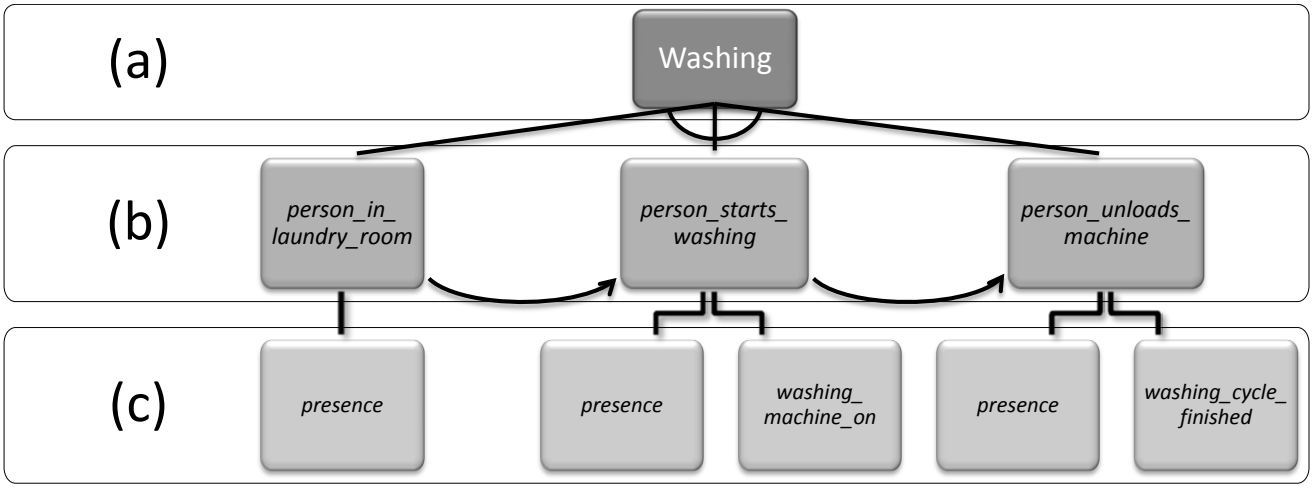


Figure 3: *A cross-layer model of the concern washing: the root-level task, (a), is decomposed in primitive tasks (situation spaces), (b), which in turn are obtained reasoning on context attributes, (c).*

troubles doing that with his arms loaded.

We now show how our current architecture can recognize the situations going on in the house and how PHATT can provide the missing activity recognition functionality. We model the root-level task `Washing` as the ordered sequence of situation spaces depicted in Figure 3.

When John enters the laundry room, the perception layer notifies the upper layer that the context attribute *presence* for the washing room has switched to value *true*. The context spaces reasoning notifies PHATT that the situation space *person-in-laundry-room* is occurring. Thus, PHATT detects that the `Washing` root-level task may be executed by the person. When John turns on the washing machine, the context spaces reasoning infers that the situation space *person-starts-washing* is occurring, which is interpreted by PHATT as the second step in the `Washing` task execution. Now John leaves the laundry room to go to the kitchen and start cooking. PHATT will observe other situation spaces occurring, e.g. *person-using-oven* and *person-using-hotplates*, primitive tasks of a root-level task `Cooking`. As we said, the model of activity execution that underlies PHATT allows multiple root-level tasks to be part of the same explanation. Thus, PHATT will generate an explanation for the observed situation spaces that contains both the goals `Washing` and `Cooking`. Until the situation space *person-unloads-machine* is observed, PHATT will consider the goal `Washing` as still active.

In our architecture, the output of PHATT can be reflected in the exploitation layer, offering contextual information that includes both the occurring situation spaces and the active goals. The augmented appliances can then exploit this information to influence their decisions. For example, if the person is walking in the corridor towards the room and the `Washing` goal is active, the probability that the person wants to enter the laundry room is higher than the probability that the person is going somewhere else. Also, the probability that the person enters the room to use the washing machine is higher than any other activity. The augmented light in the laundry room is notified with this contextual information and prepares to switch itself on as soon as the person actually enters the room. This is useful because other situations and activities may not require turning on the light. For instance, a person may just want to enter the laundry room, grab something and then exit without turning on the light. Knowing that the `Washing` goal is active, instead, helps deciding that the person may need enough light to operate the washing machine. In this way, the decisions taken by the augmented appliances to manage their behavior are helped by a double level of information: the situation spaces, obtained by statically analyzing the sensing data coming from the physical environment, and the root-level goals, produced by PHATT by analyzing the dynamics of situation spaces, generating all the possible explanations and evaluating their probabilities.

5 Open Issues

To conclude this paper, we present some aspects that we will investigate in future work.

In Sect. 4, we provided an example of domestic activity scenario. As we highlighted, that scenario is not representative of the complex model of activity that we presented in this paper.

For instance, the case of multiple people acting and interacting is not taken into account by the proposed scenario. In particular, the person that enters the laundry room may not be the one that is in charge of the laundry. Or, alternatively, the person may want to enter the laundry room for a different reason than doing the laundry. As we explained, taking into account these eventualities is important since it is impossible to model and reproduce what happens in inhabitants' minds. For this reason, future work will have to address the issue of non-interruptive takeover of the system. For instance, if the light automatically turns on in the laundry room, the person should be provided with a proximate interface to turn it off

if preferred.

Another aspect of activity that we plan to consider shortly is the recurrence of concerns. Even though the time or order of actualization of concerns cannot be predicted, we can indeed take into consideration the recurrent nature of some of them. For this purpose, the concepts of *prior goal probability* and of influence of the *state of the world* introduced in (Goldman, Geib, and Miller 1999) look very promising.

The recognition of complex ambiguous behaviors of inhabitants could be improved following the principles described in (Coutaz et al. 2005), combining PHATT with the context spaces and with the perception layer using a *holistic* approach. In other words, we may use PHATT to provide feedback to the underlying context spaces reasoning and sensor data fusion layers. The feedback could be positive or negative, depending on the output of PHATT: if the top-ranked explanation has a high probability with respect to the others and as an absolute value, we may return to the lower layers a positive feedback. This feedback allows confirming the results of the context reasoning process and strengthening the belief in the sensor data fusion results, for instance exploiting the *conditioning* function described in (Smets and Kruse 1996).

Concerning the implementation aspects of PHATT, we need to carefully consider some important aspects, described in the rest of this Section.

(Geib and Goldman 2009) presents the assumption that each goal of an agent is known since the beginning of the execution. In Sect. 3, we said that human activity is characterized by inter-individual interactions and interactions with the physical environment, which result in an "on-the-fly" modification of the concerns the person is involved in. The assumption made by PHATT's implementation is thus clearly in contrast with our model of activity. Practically, the consequence of this assumption is that all the explanations and the probabilities have to be recalculated when a new goal is discovered. Future work will investigate the consequences of removing this assumption.

PHATT is implemented with the underlying assumption that all the actions performed by the agent are either directly observable or, in the case of *adversarial plan recognition* (Geib and Goldman 2001), inferable from their effects (changes in the *state of the world* or observation of "disabled" actions (Geib and Goldman 2001)). No concept of degree of uncertainty in the observations is considered, so there is no way to take into consideration the confidence value provided as output of the context spaces reasoning process. Further work will study ways to incorporate the confidence measure into PHATT or effective ways to choose confidence thresholds able to discern the occurrence of situation spaces.

We also need to investigate how the same situation space can be part of different root-level tasks. In (Goldman, Geib, and Miller 1999), the modeling allows the same primitive task to belong to multiple root-level tasks (the authors call *overloaded* these primitive tasks (Goldman, Geib, and Miller 1999)). In the implementation described in (Geib and Goldman 2009), this aspect is left as an open research question. In our system, different human concerns can reflect in the detection of the same situation spaces, and the same concern can be actualized in different situation spaces, leading to the need to model overloaded primitive tasks.

6 Conclusions

In this paper, we showed that the limitations of existing activity recognition approaches in the smart home domain are often due to the poverty of the adopted human-activity models. Combining expertise from the cognitive ergonomics and the ubiquitous computing fields, we discussed the hard technical challenges to address when leveraging on a realistic model of human activity. We presented the architecture of a smart home prototype that we are developing and showed the gap that exists between our current capabilities in terms of contextual-knowledge extraction and the complexity of the targeted activity recognition. To fill this gap, we proposed and discussed the integration of an existing algorithm for plan recognition into our system, in order to mine additional information from the dynamics of context.

Much work is still needed to perform activity recognition when accepting the difficult challenges raised by our complex naturalistic human activity model. Our ultimate aim is to extract a rich basis of contextual information to be used to provide the inhabitants of a smart home with adapted functionalities, targeted to obtain energy saving while preserving inhabitants' comfort.

References

Aldrich, F. 2003. Smart homes: Past, present and future. In Harper, R., ed., *Inside the Smart Home*. Springer London. 17–39.

Baillie, L., and Benyon, D. 2008. Place and technology in the home. *Computer Supported Cooperative Work (CSCW)* 17:227–256. 10.1007/s10606-007-9063-2.

Bonhomme, S.; Campo, E.; Esteve, D.; and Guennec, J. 2008a. Methodology and tools for the design and verification of a smart management system for home comfort. In *Intelligent Systems, 2008. IS '08. 4th International IEEE Conference*, volume 3, 24–2 –24–7.

Bonhomme, S.; Campo, E.; Esteve, D.; and Guennec, J. 2008b. PROSAFE-extended, a telemedicine platform to contribute to medical diagnosis. *J Telemed Telecare* 14(3):116–119.

Boytsov, A.; Zaslavsky, A.; and Synnes, K. 2009. Extending context spaces theory by predicting run-time context. In *NEW2AN '09 and ruSMART '09: Proceedings of the 9th International Conference on Smart Spaces and Next Generation Wired/Wireless Networking and Second Conference on Smart Spaces*, 8–21. Berlin, Heidelberg: Springer-Verlag.

Campo, E.; Bonhomme, S.; Chan, M.; and Esteve, D. 2006. Learning life habits and practices: an issue to the smart home. In C.Nugent, J., ed., *Smart homes and beyond, ICOST'2006 - 4st International Conference On Smart homes and health Telematic*, 355–358. Belfast: IOS Press.

Chahuara, P.; Vacher, M.; and Portet, F. 2010. Localisation d'habitant dans un environnement perceptif non visuel par propagation d'activation multisource. In *MAJECSTIC*, 8pp.

Chan, M.; Estve, D.; Escriba, C.; and Campo, E. 2008. A review of smart homes–present state and future challenges. *Computer Methods and Programs in Biomedicine* 91(1):55 – 81.

Chen, L., and Nugent, C. D. 2009. Ontology-based activity recognition in intelligent pervasive environments. *IJWIS* 5(4):410–430.

Coutaz, J.; Crowley, J. L.; Dobson, S.; and Garlan, D. 2005. Context is key. *Commun. ACM* 48:49–53.

Crabtree, A., and Rodden, T. 2004. Domestic routines and design for the home. *Comput. Supported Coop. Work* 13:191–220.

Geib, C. W., and Goldman, R. P. 2001. Plan recognition in intrusion detection systems. *DARPA Information Survivability Conference and Exposition,* 1:0046.

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

Ghallab, M.; Nau, D.; and Traverso, P. 2004. Hierarchical task network planning. 229 – 261.

Goldman, R. P.; Geib, C. W.; and Miller, C. A. 1999. A New Model of Plan Recognition. *Artificial Intelligence* 64:53–79.

Greenberg, S. 2001. Context as a dynamic construct. *Hum.-Comput. Interact.* 16:257–268.

Gu, T.; Wang, X. H.; Pung, H. K.; and Zhang, D. Q. 2004. An ontology-based context model in intelligent environments. 270–275.

Gu, T.; Wu, Z.; Tao, X.; Pung, H. K.; and Lu, J. 2009. epSICAR: An Emerging Patterns based approach to sequential, interleaved and Concurrent Activity Recognition. In *Proc. IEEE International Conference on Pervasive Computing and Communications PerCom 2009*, 1–9.

Guibourdenche, J.; Vacherand-Revel, J.; Grosjean, M.; Frjus, M.; and Haradji, Y. 2011. Using multiple scores for transcribing the distributed activities of a family. In *Proceedings of the 2011 ACM conference on Computer supported cooperative work*, CSCW '11. New York, NY, USA: ACM.

Kim, E.; Helal, S.; and Cook, D. 2010. Human activity recognition and pattern discovery. *IEEE Pervasive Computing* 9:48–53.

La Placa, M.; Pigot, H.; and Kabanza, F. 2009. Assistive planning for people with cognitive impairments. In *Proc. of Workshop on Intelligent Systems for Assisted Cognition hosted by Int'l Joint Conference on Artificial Intelligence (IJCAI)*.

Padovitz, A.; Loke, S.; Zaslavsky, A.; and Burg, B. 2007. Verification of uncertain context based on a theory of context spaces. *International Journal of Pervasive Computing and Communications* 3(1):30–56.

Padovitz, A.; Zaslavsky, A.; and Loke, S. W. 2006. A unifying model for representing and reasoning about context under uncertainty. In *Proceedings of the 11th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU)*.

Padovitz, A. 2006. *Context Management and Reasoning about Situations in Pervasive Computing*. Ph.D. Dissertation, Monash University, Australia.

Poizat, G.; Fréjus, M.; and Haradji, Y. 2009. Analysis of activity in domestic settings for the design ubiquitous technologies. In *European Conference on Cognitive Ergonomics: Designing beyond the Product — Understanding Activity and User Experience in Ubiquitous Environments*, ECCE '09, 14:1–14:2. VTT, Finland, Finland: VTT Technical Research Centre of Finland.

Ricquebourg, V.; Delafosse, M.; Delahoche, L.; Marhic, B.; Jolly-Desodt, A.; and Menga, D. 2007. Fault Detection by Combining Redundant Sensors: a Conflict Approach Within the TBM Framework. In *COGIS 2007, COGnitive systems with Interactive Sensors*. Stanford University.

Roy, P. C.; Bouchard, B.; Bouzouane, A.; and Giroux, S. 2010. *Web Intelligence and Intelligent Agents*. InTech. chapter Combining Pervasive Computing with Activity Recognition and Learning, 447–462. ISBN: 978-953-7619-85-5.

Salembier, P.; Dugdale, J.; Frejus, M.; and Haradji, Y. 2009. A descriptive model of contextual activities for the design of domestic situations. In *European Conference on Cognitive Ergonomics: Designing beyond the Product — Understanding Activity and User Experience in Ubiquitous Environments*, ECCE '09, 13:1–13:7. VTT, Finland, Finland: VTT Technical Research Centre of Finland.

Shelby, Z., and Bormann, C. 2009. *6LoWPAN: The Wireless Embedded Internet*. John Wiley & Sons, Ltd.

Smets, P., and Kruse, R. 1996. The transferable belief model for belief representation. In *Uncertainty Management in Information Systems*. Boston: Kluwer Academic Publishers. 343–368.

Theureau, J. 2003. *Handbook of cognitive task design*. New Jersey: Lawrence Erlbaum Associates. chapter Course-of-action analysis and course-of-action-centered design, 55–81.

Weiser, M. 1993. Some computer science issues in ubiquitous computing. *Commun. ACM* 36:75–84.

New Algorithms and Hardness Results for Multi-Agent Plan Recognition

Bikramjit Banerjee
School of Computing
The University of Southern Mississippi
Hattiesburg, MS 39406

Jeremy Lyle
Dept. of Mathematics
The University of Southern Mississippi
Hattiesburg, MS 39406

Landon Kraemer
School of Computing
The University of Southern Mississippi
Hattiesburg, MS 39406

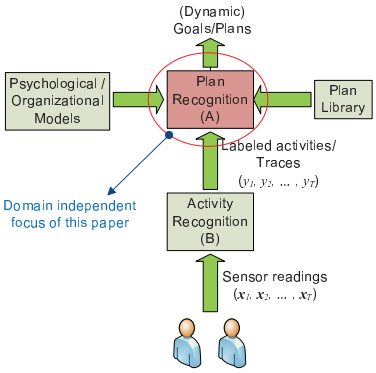
Abstract

We extend our recent formalization of multi-agent plan recognition (MAPR), to accommodate compact multi-agent plan libraries and incomplete plans, and propose polynomial time algorithms for several cases of static teams: when team-size is bounded by 2, or when the social structure graph is a star, a tree of bounded depth, or a path. However, we show that when the teams are dynamic and even when the social structure graph is as simple as a path, MAPR is NP-complete. Finally, we show rigorously for the first time, that when activity interleaving is allowed, even the single agent version of MAPR is NP-complete.

Introduction

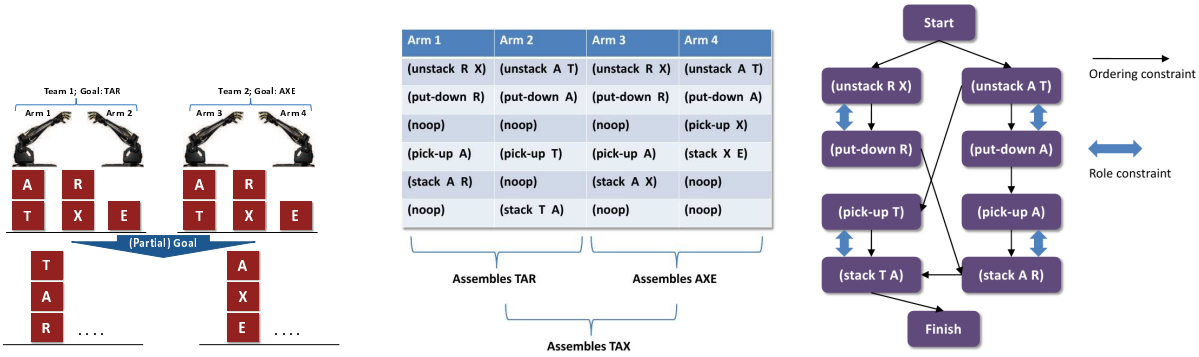
Multi-agent plan recognition (MAPR) refers to the problem of explaining the observed behavior of multiple agents by identifying the (dynamic) team-structures and the team plans (based on a given plan library) being executed, as well as predicting their future behavior. Recently, we introduced a formal model for MAPR and used it to investigate the complexity of its simplest setting (Banerjee, Kraemer, and Lyle 2010). However, this model has several limitations which we address in this paper, and investigate the complexities of various settings in a richer model.

Our focus is on the symbolic MAPR problem, as shown below, in order to develop MAPR theory in a *domain-independent* way. As such, we abstract away the complex problem of sensor interpretation (activity recognition), to wean MAPR out of domain-dependency, and assume that a symbolic trace and a plan library are available in a common language. We begin with an illustration of this abstracted MAPR problem in a multi-agent blocks words domain, shown in Figure 1. In part (a), two teams of robotic arms assemble the goal words "TAR" and "AXE" from separate stacks, starting from the (not necessarily) same initial configuration. Part (b) shows the trace of 6 steps of activities of the 4 robotic arms, as seen by the (remote) recognizer. The recognizer works with incomplete information, i.e., the association between the arms and the stack identifiers (that would have enabled it to identify teams directly) are unavailable. Therefore, while arms 1 and 2 jointly assemble "TAR", and arms 3 and 4 jointly assemble "AXE", arms 2 and 3 appear to assemble "TAX" as well, creating ambiguity for the recognizer. The key insight is that it is impossible to *partition* the trace into non-overlapping, complete or incomplete team-plans if the goal hypothesis "TAX" is accepted. Note, teammates are not required to start plan execution at the same time, and may not complete a plan by the observation horizon, making probabilistic prediction a useful objective. Part (c) shows a (non-unique) plan from the library, for start state in (a) and goal "TAR", in the form of a plan graph. This is a graph based on the partially ordered set of steps needed to achieve a goal from a start state, with added constraints: *role constraints* (which steps need to be performed by the same agent) and *concurrency constraints* (which steps need to be executed simultaneously; not needed in this illustration). Note, the duration and the team size needed to execute a plan are unspecified though constrained, e.g., 1 to 4 agents can execute this plan in 5 to unlimited time steps (due to noops).



Typically for plan recognition with single agents, a plan library is given in a compact hierarchical form, such as an HTN (Erol, Hendler, and Nau 1994). Formulating such a library for a multi-agent system is more complex (Sukthankar and Sycara 2008). In this paper, we develop algorithms and complexity results for two less expressive (than HTNs) plan libraries, viz., context free grammars (CFGs) and plan graphs (Figure 1(c)), each of which incorporates some desirable features of HTN, e.g., recursiveness and hierarchies in CFGs, and partial ordering in plan graphs. Both advance our previous formalization in (Banerjee, Kraemer, and Lyle 2010) which accommodated none of these desirable features.

In this paper, we refine our previous model (Banerjee,



(a) (b) (c)

Figure 1: Multi-Agent Blocks Words.

Kraemer, and Lyle 2010) to make 2 important generalizations: allow compact, non-trivial plan libraries that correspond to infinite languages as opposed to the finite language in (Banerjee, Kraemer, and Lyle 2010), and relax the assumption in (Banerjee, Kraemer, and Lyle 2010) that all observed plans are completed by the observation horizon. We propose a multi-agent context free grammar to compactly describe multi-agent plans, and adapt Vilain's Earley-based algorithm (Vilain 1990) to parse multi-agent activity strings with this grammar to yield the highest valued parse. We use this algorithm to polynomially solve several special cases of MAPR when the teams are static: where team-size is bounded by 2, or where the social structure graph is a star, a tree of bounded depth, or a path. However, when the teams are dynamic, we show that even when the social structure graph is as simple as a path, and the library contains plan graphs, MAPR is NP-complete. Finally, we show rigorously for the first time, that when activity interleaving is allowed, even a single agent MAPR is NP-complete.

Preliminaries

Let A be a set of n agents, $\{a_1, a_2, \ldots, a_n\}$, and Σ be a fixed size alphabet of grounded, primitive actions (e.g., "(unstack A T)") that these agents can be observed to execute. We are given a trace, $\mathcal{T}$, of observed activities of these agents over T steps of time, in the form of a $T \times n$ matrix, $\mathcal{T} = [t_{ij}]$, where $t_{ij} \in \Sigma$ is the action executed by agent a_j at time i, $j = 1, \ldots, n$ and $i = 1, \ldots, T$. Note that we actually do not require the observed agents to act in a synchronized manner, as Figure 1(b) may suggest. Rather, sensor interpretations are reported with the timestamps of the corresponding observations, which are then placed into the trace rows on a discretized time scale. The resolution of this discretization is such that no two symbolic activities of any agent fall into the same trace cell, i.e., the resolution of the trace rows is adapted to the fastest agent, with "(noop)"s filling the resulting empty cells of the slower agents.

We are also given a finite library of team plans $\mathcal{L}$, in some form. In this paper, our choice of a representation for the plan library is guided by a need to strike a middle ground between polynomial solvability of some cases, against the NP-completeness of others. Our strategy is to select a more expressive language (in particular, context free grammars) for the easier cases, but a more limited language (in particular, the finite language from (Banerjee, Kraemer, and Lyle 2010)) for the harder cases. The rationale for such a strategy is that the results are expected to remain unchanged (or become easier and harder respectively) when more restrictive representations (such as special cases of context free grammars) are considered for the easier cases, or more general representations (such as the infinite language corresponding to the plan graph representation introduced above) for the harder cases. We first introduce the various forms of the library for MAPR.

The Plan Library

We define $\mathcal{L}$ in three different forms. The first is *context free grammar* to be used to parse strings of activity vectors of agents. Following the conventions of (Kautz and Allen 1986; Vilain 1990), we assume that all plans are either END plans or not. An END plan is one that is meaningful in and of itself, while a non-END plan can only occur as a component of some other plan. We define a set of END goals, such that each END plan derives some END goal. As in (Vilain 1990), the start production rule is

$$S \rightarrow END \mid END\ S$$

The production for all END goals is given by

$$END \rightarrow P_1^j \mid P_2^{j'} \mid \ldots \quad (1)$$

where P_i^j is the ith END goal, that can be achieved by a team containing j agents. The production of P_i^j, which represents an END plan, takes the following form

$$P_i^j \rightarrow \ldots Q_1^j \ldots Q_2^j \ldots$$

where the right hand side of the rule contains non-END non-terminals Q_i^j that only describe (sub)goals of j-agent teams. All terminals on the right hand side above are also j length vectors of symbols from Σ. Moreover, the productions of Q_i^j are also limited to j length terminals and j length non-END non-terminals Q_k^j. Additionally, we require that no END

goal, P_i^j, ever appears on the right hand side of any rule except rule 1. In other words, an END goal P_i^j cannot be a part of itself or another END goal P_k^j. This is a technical requirement, and is not truly an assumption, See (Banerjee, Lyle, and Kraemer 2011) for the justifications of these assumptions.

Allowing the capability of recursion enables us to compactly represent a plan such as *bounding overwatch*, which could be represented by the following rules

$$\begin{aligned} Q^j &\rightarrow X^jY^j \mid X^jY^jQ^j \\ X^j &\rightarrow (fire^k, withdraw^{j-k}) \mid (fire^k, withdraw^{j-k})X^j \\ Y^j &\rightarrow (withdraw^k, fire^{j-k}) \mid (withdraw^k, fire^{j-k})Y^j \end{aligned}$$

If Q^j must be an END goal, we simply add a dummy END goal P^j and a dummy END plan $P^j \rightarrow Q^j$, to satisfy the technical requirement.

The second form of $\mathcal{L}$ is a *finite collection of plan graphs*, used in the illustration in Figure 1(c). Plan graphs grounded in start-goal states can be viewed as being produced by *decomposition* (Ghallab, Nau, and Traverso 2004) from a (more abstract and traditional) Hierarchical Task Network (Erol, Hendler, and Nau 1994) plan library into a partially ordered set of primitive actions (which we call the plan graph), after a team of agents have chosen a goal. Notice, we do not assume that all agents must start their team activities at the same time (as the illustration in Figure 1 (b) might suggest) since agents can include arbitrary numbers of "noop"s before and between operators.

A third form for $\mathcal{L}$ is a finite collection of finite matrices of symbols from Σ. This library, that we used before in (Banerjee, Kraemer, and Lyle 2010), is hardly practical, but its purpose is to establish *baseline* hardness results such that more practical libraries are likely to make those cases even harder. It is straightforward to see that the third language above is the least expressive and is a special case of the other two, since it corresponds to highly constrained plan graphs, and can also be expressed by a regular grammar – a special case of CFG. Therefore, the first two forms of $\mathcal{L}$ can be seen as engendering a set of matrices, but this set can be infinite. Thus hardness results based on the third language should also carry forward to the other two languages. For polynomial solvability, however, the results would be more interesting with the CFG library.

Definitions

As mentioned before, we assume the library to be in different form for different cases, but for the sake of uniformity let $\mathcal{L} \xrightarrow{\pi} p$ denote the fact that an $x \times y$ matrix of symbols from Σ, say p, is engendered by some plan π in the library $\mathcal{L}$. We do not require x to be related to T, or y to n. Formally,

Definition 1. **($\xrightarrow{\pi}$)** *Given an $x \times y$ matrix p, $p_{ij} \in \Sigma$, we say $\mathcal{L} \xrightarrow{\pi} p$ iff*

- *$\mathcal{L}$ can derive p using a top level production rule π, when $\mathcal{L}$ is a context free grammar,*
- *p satisfies all the ordering, role and concurrency constraints of $\pi \in \mathcal{L}$, when $\mathcal{L}$ is a finite set of plan graphs (Figure 1(c))*
- *$p = \pi$, when $\mathcal{L}$ is a finite collection of matrices of symbols from Σ.*

The $x \times y$ matrix p above can be thought of as the trace of activities of one team of y agents, for x steps. In the rest of the paper, we shall represent the number of rows of a matrix p as $r(p)$ and the number of its columns as $c(p)$. The above definition connects a p to the plans in the library $\mathcal{L}$. The following definition connects it to the trace $\mathcal{T}$, in which case it is necessary that $y \leq n$, but the correspondence between the columns of p and the set A (of agents) is unspecified.

Definition 2. (Occurrence) *An occurrence of a matrix p (of symbols from Σ, and with $\leq$ n columns and a finite number of rows) in the trace $\mathcal{T}$ is given by a tuple $o_p = (k_1, k_2, \ldots, k_{c(p)}, t_p)$ such that*

- $1 \leq t_p \leq T$
- $a_{k_j} \in A$ *and* $k_i \neq k_j,\ 1 \leq i, j \leq c(p)$
- $p_{ij} = \mathcal{T}(t_p + i - 1, k_j), i = 1, \ldots, \tau,\ j = 1, \ldots, c(p)$

where $\tau = \min\{r(p), T - t_p + 1\}$. In other words, if τ contiguous rows, (viz., $t_p, t_p + 1, \ldots, t_p + \tau - 1$), and $c(p)$ columns (say $k_1, \ldots, k_{c(p)}$, a $c(p)$-selection in any order from n agent indices) can be found in $\mathcal{T}$ such that the resulting submatrix exactly matches the $\tau \times c(p)$ (sub)matrix of p, then p occurs in $\mathcal{T}$. If $\tau = r(p)$, then the occurrence is complete, but if $r(p) > T - t_p + 1$ then the occurrence is partial.

A partial occurrence can be interpreted as yielding a *prediction* of what observations can be expected beyond the observation horizon, T. Since it is not guranteed that all observed plans will have completed by the observation horizon T, allowing partial occurrences is an important generalization of (Banerjee, Kraemer, and Lyle 2010). Furthermore, such predictions are useful since they can help validate (or revise) the current explanations when more observations become available.

Note that a given p can have multiple occurrences in $\mathcal{T}$. Two occurrences of p in $\mathcal{T}$, $(k_1, k_2, \ldots, k_{c(p)}, t_p)$ and $(k_1^{'}, k_2^{'}, \ldots, k_{c(p)}^{'}, t_p^{'})$, are distinct iff $t_p \neq t_p^{'}$ or $\{k_1, k_2, \ldots, k_{c(p)}\} \neq \{k_1^{'}, k_2^{'}, \ldots, k_{c(p)}^{'}\}$. We represent the set of all distinct occurrences of p in $\mathcal{T}$ as $\mathcal{O}_{p,\mathcal{T}}$.

In order to formalize the partitioning of $\mathcal{T}$ using various occurrences, we first formalize the notion of *conflict* of two occurrences in the following definition.

Definition 3. (Conflict) *Two occurrences of matrices p, q (same or distinct, partial or complete), $o_p = (k_1, k_2, \ldots, k_{c(p)}, t_p)$, $o_q = (k_1^{'}, k_2^{'}, \ldots, k_{c(q)}^{'}, t_q)$ are said to be in conflict iff both of the following hold:*

- $\{k_1, k_2, \ldots, k_{c(p)}\} \bigcap \{k_1^{'}, k_2^{'}, \ldots, k_{c(q)}^{'}\} \neq \emptyset$
- $t_p \leq t_q + r(q) - 1$ *and* $t_q \leq t_p + r(p) - 1$

Finally, a partition of the trace $\mathcal{T}$ for a given library $\mathcal{L}$ is defined as follows:

Definition 4. (Partition) *A partition of $\mathcal{T}$ for a given library $\mathcal{L}$, represented as $\Pi_{\mathcal{T}|\mathcal{L}}$, is a set of triples, (p, o_p, π), such that all of the following hold:*

- $o_p \in \mathcal{O}_{p,\mathcal{T}},\ \forall (p, o_p, \pi) \in \Pi_{\mathcal{T}|\mathcal{L}}$,
- *For each* $(p, o_p, \pi) \in \Pi_{\mathcal{T}|\mathcal{L}}$, $\mathcal{L} \xrightarrow{\pi} p$,
- *There is no pair of triples,* $(p, o_p, \pi_p), (q, o_q, \pi_q)$ *in* $\Pi_{\mathcal{T}|\mathcal{L}}$, *such that* o_p, o_q *are in conflict,*
- *For each* (i, j) *such that* $1 \leq i \leq T,\ 1 \leq j \leq n$, *there exists* $(p, (k_1, \ldots, k_p, t_p), \pi_p) \in \Pi_{\mathcal{T}|\mathcal{L}}$ *such that* $t_p \leq i \leq t_p + r(p) - 1$ *and* $j \in \{k_1, \ldots, k_p\}$.

We call the set of possible partitions (whose finiteness depends on the nature of $\mathcal{L}$) of $\mathcal{T}$, $\mathcal{P}$. We associate a utility function $f : \mathcal{P} \mapsto \Re$ to the partitions, so that each partition of $\mathcal{T}$ can be evaluated for its preferability as an explanation for the activities observed, as well as possible predictions of some activities beyond T (when occurrences are partial). We can now define the MAPR problem as follows:

Definition 5. (MAPR) *The multi-agent plan recognition problem, represented as MAPR($\mathcal{T}_{T\times n}, \mathcal{L}, f, k$) is defined as follows:*

Instance: *A fixed set of symbols,* Σ*; activity matrix* $\mathcal{T}$ *(of size* $T \times n$*) such that* $t_{ij} \in \Sigma$*, a plan library* $\mathcal{L}$*, a function* $f : \mathcal{P} \mapsto \Re$ *and* $k \in \mathbb{Z}$.

Decision Question: *Is there a partition,* $\Pi_{\mathcal{T}} = \{(p, o_p, \pi), \ldots\}$ *of* $\mathcal{T}$ *such that* $f(\Pi_{\mathcal{T}|\mathcal{L}}) \geq k$*?*

Optimization Question: *Which partition of* $\mathcal{T}$*, if any, say* $\Pi_{\mathcal{T}|\mathcal{L}} = \{(p, o_p, \pi), \ldots\}$ *maximizes* $f(\Pi_{\mathcal{T}|\mathcal{L}})$*? We represent the optimization problem as MAPR($\mathcal{T}_{T\times n}, \mathcal{L}, f$).*

The objective of interest is non-unique (Banerjee, Lyle, and Kraemer 2011), but we only consider the optimization problem here.

The Utility Function

The number of possible partitions may not be a polynomial in n, T, or even finite; consequently the size of the function f may not be compact. Without assuming some kind of structure in f, it may be hard to ensure its polynomial computability, without which polynomial time MAPR appears hopeless. As in (Banerjee, Kraemer, and Lyle 2010), we assume f is *additive* for polynomial time results, and of the form

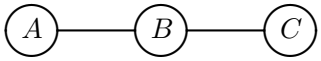
$$f(\{(p_1, o_{p_1}, \pi_1), \ldots, (p_z, o_{p_z}, \pi_z)\}) = \sum_i v(p_i, o_{p_i}, \pi_i),$$

v being some value function that maps the triples (p_i, o_{p_i}, π_i) to values. See (Banerjee, Lyle, and Kraemer 2011) for more details.

Social Structures

One of the major goals of this paper is to present polynomial solvability results for some interesting special cases of MAPR, where special structures are exploited. In the past, *social structures*, i.e., some known organizational structure among the observed agents have been used for solving MAPR but such studies have been constrained by very specific application domains (Kaminka, Pynadath, and Tambe 2002; Tambe 1996).

We consider social structures given by graphs, where agents are the vertices. We say that agents that constitute a path in this graph can form a team, but not otherwise. This prevents agents from "jumping hierarchy" and also captures the notion of a team leader in a hierarchical setting. In fact, we consider the hierarchical social structure given by a tree of a bounded depth. This is a practical consideration, since in reality the number of levels in a hierarchy are often bounded, but the number of members can be variable. We also consider more restricted social structures such as a star (tree hierarchy of depth 1) and path (special tree of variable depth, representing a "chain of command") graphs. For instance, for the following path graph on 3 agents, $\{A, B, C\}$, the possible teams are $\{\{A\}, \{B\}, \{C\}, \{A, B\}, \{B, C\}, \{A, B, C\}\}$, but $\{A, C\}$ is not a valid team.



A social structure graph prevents *arbitrary* teams, and thus imposes structure on MAPR to allow us to solve some special cases easily. In particular for path graphs, since the number of possible teams is rendered polynomial, MAPR can be solved in polynomial time if the teams are static. However, even with a polynomial number of possible teams, if the teams can change dynamically, we show later that MAPR is NP-complete. In the context of social structures defined above, our previous hardness result (Banerjee, Kraemer, and Lyle 2010) can be interpreted as being based on a social structure graph that is *complete* thus allowing arbitrary (i.e., an exponential number of) teams.

Non-interleaved Plan Execution

The problem formulation in the Definitions section does not accommodate interleaved plan execution by the observed agents. In other words, an agent must complete a plan before moving on to a different plan. All of our results in this section fall in the non-interleaved category, but later in the paper we present the first rigorous hardness result for MAPR in the face of interleaved plan execution. We consider both static and dynamic teams.

Static Teams

In many situations, the team structure among the agents may remain static through the observation horizon T. This is clearly the case, for instance, in several application domains where multi-agent activity recognition has been explored, such as multi-robotic soccer (Vail and Veloso 2008) and multi-agent capture-the-flag games (Sadilek and Kautz 2010). Interestingly, MAPR is NP-complete even if the teams are static but can be of size 3 or more, as our proofs in (Banerjee, Kraemer, and Lyle 2010) demonstrate. However, it is unknown if additional structure in the form of a bound on the team size or a known social structure can be exploited to solve MAPR more easily. In this section we show that if the team size is bounded by 2, or if the social structure is a star, path, or a bounded-depth tree, then MAPR is in P.

Algorithm 1 ROLECOMBINE(r, q)

1: **Input:** Two vectors of sets of integers r and q, both of length s
2: $z_i \leftarrow r_i \cap q_i$ **for** $i \leftarrow 1, 2, \ldots, s$
3: **for** $i \leftarrow 1, \ldots, s$ **do**
4: $w_i \leftarrow \{j \mid 1 \leq j \leq s, z_i = z_j\}$
5: **if** $|w_i| \neq |z_i|$ **then**
6: **Return** $\emptyset$
7: **end if**
8: **end for**
9: **Return** z

All of the polynomial time results in this section are based on the CFG plan library introduced earlier, for which we present algorithm PARSE, derived from Vilain's Earley-based parser (Vilain 1990). The input to this algorithm are: an $L \times s$ matrix x of symbols from Σ representing the activities of s agents for L steps; a context-free grammar $\mathcal{L}$ with a set of top-level non-terminals of team-size s only $P = \{P_1^s, P_2^s, \ldots, P_w^s\}$ representing END-goals; and an occurrence o_x.

The output of PARSE is the highest valued partition of x and the corresponding value ($(\emptyset, *)$ if the parse fails). In Algorithm 2, α represents an arbitrary terminal (vector of length s), β an arbitrary terminal or non-terminal, and uppercase unitalicized letters represent arbitrary non-terminals. o_{xi} represents the occurrence o_x with the start time replaced by i. $x_{i,j}$ represents the jth element of the vector x_i of length s, and x_i^k represents the submatrix of x from rows i thru k (and all s columns).

Our parser is adapted for multi-agent CFG (i.e., vector terminals instead of scalar) presented in the Plan Library section, and to accommodate a value function and partial occurrences. Steps 1–34 are same as Earley's parser with predict-scan-complete loop, except steps 19–22 which help maintain a chain of END rules that had the best parse value, completing at observation k. This allows us to return the *highest valued* parse in contrast to Earley. Besides, in order to ensure consistency of agent roles from one terminal to the next in a parse of the *same* END goal, we maintain the role hypothesis (which agent column in a terminal matches which column of matrix x) as an additional part of $states$ (basic Earley parser only maintains the dotted rule and the start index), and use the function ROLECOMBINE (Algorithm 1) to verify if two role hypotheses r and q are consistent, and if so, return a combined hypothesis ($\emptyset$ otherwise).

Step 35 is a repeat of the "Complete" block (lines 18–32), but only on $states[L]$ and is an addition to the Earley parser, to accommodate the values of partial occurrences into the dynamic programming. For every incomplete END-plan that started at or before L, we *fake* completion (i.e., we do not advance the $\bullet$ as in line 29) and see if a complete parse upto j followed by a partial occurrence till L can give us a better partition of x_1^L. In other words, we solve the problem $V(L) = \max_{1 \leq j < L}[V(j) + bestpartial(x_{j+1}^L)]$ by dynamic programming. If the maximum number of dotted rules afforded by the grammar is G, then since there are L steps of observation, s agents, and ROLECOMBINE is $O(s^3)$, step 35

Algorithm 2 PARSE$(x, \mathcal{L}, o_x)$

1: **Initialize:** $bestpartition\,[0 \ldots L] \leftarrow [\ \emptyset \ \ \emptyset \ \ \ldots \ \ \emptyset\ ]$; $bestval\,[0 \ldots L] \leftarrow [\ 0 \ \ 0 \ \ \ldots \ \ 0\ ]$; $states\,[0 \ldots L] \leftarrow [\ \emptyset \ \ \emptyset \ \ \ldots \ \ \emptyset\ ]$; $roles\,[0 \ldots s] \leftarrow [\ \{1, 2, \ldots, s\} \ \ \ldots \ \ \{1, 2, \ldots, s\}\ ]$
2: Add (S $\rightarrow \bullet$END S$, 0, roles$) **and** (S $\rightarrow \bullet$END$, 0, roles$) to $states[0]$
3: **for** $k \leftarrow 0$ **to** L **do**
4: **repeat**
5: ——————Predict——————
6: **for all** (Q $\rightarrow \ldots \bullet$ Y $\ldots, i, r) \in states[k]$ **do**
7: add (Y $\rightarrow \bullet\beta \ldots, k, roles$) to $states[k]$ for all productions in $\mathcal{L}$ with Y on the lefthand side.
8: **end for**
9: ——————Scan——————
10: **for all** (Q $\rightarrow \ldots \bullet\, \alpha \ldots, i, r) \in states[k]$ **do**
11: $r'_m \leftarrow \{j | 1 \leq j \leq s, x_{k+1,m} = \alpha_j\}$ **for** $m \leftarrow 1, 2, \ldots, s$
12: $z \leftarrow$ROLECOMBINE(r, r')
13: **if** $z \neq \emptyset$ **then**
14: add (Q $\rightarrow \ldots \alpha \bullet \ldots, i, z$) to $states[k+1]$
15: **end if**
16: **end for**
17: ——————Complete——————
18: **for all** (Q $\rightarrow \ldots \beta\bullet, i, r) \in states[k]$ **do**
19: **if** Q $\in P$ and ($bestpartition_k = \emptyset$ or $bestval_k < bestval_i + v(x_i^k, o_{xi}$, Q $\rightarrow \ldots \beta)$) **then**
20: $bestpartition_k \leftarrow concat(bestpartition_i$, (Q $\rightarrow \ldots \beta, i+1, k))$
21: $bestval_k \leftarrow bestval_i + v(x_i^k, o_{xi}$, Q $\rightarrow \ldots \beta\bullet)$
22: **end if**
23: **for all** (R $\rightarrow \ldots \bullet$ Q$\ldots, j, r') \in states[i]$ **do**
24: $z \leftarrow$ROLECOMBINE(r, r')
25: **if** Q $\in P$ **then**
26: $z \leftarrow roles$
27: **end if**
28: **if** $z \neq \emptyset$ **then**
29: add (R $\rightarrow \ldots$ Q $\bullet \ldots, j, z$) to $states[k]$
30: **end if**
31: **end for**
32: **end for**
33: **until** no states can be added to $states[k]$
34: **end for**
35: Repeat the "Complete" block above with k set to L, but only on incomplete rules (line 18) and do not advance $\bullet$ (line 29).
36: **Return** $(bestpartition_L, bestval_L)$

is $O(s^3GL)$. Therefore, the complexity of the entire algorithm is still the same as Earley's with the additional factor of s^3 to account for vector terminals, leading to a complexity of $O(s^3G^2L^3)$. Since RoleCombine is rather simple and Parse is a simple extension of (Vilain 1990), we omit proofs of their correctness. Based on Parse, we give the following results (proofs can be found in (Banerjee, Lyle, and Kraemer 2011)):

Theorem 1. *When $\mathcal{L}$ is a context free grammar and team-sizes are bounded by 2, then MAPR($\mathcal{T}_{T\times n}, \mathcal{L}, f$) can be solved in time $O(n^{2.5}G^2T^3)$ for additive f.*

We now consider how a known social structure can be exploited to solve MAPR in polynomial time, even when the (static) teams can have more than two agents. We first present the result when the social structure graph is a *star*, which forms the base case for induction on multi-level trees of bounded depth. Finally, we consider a path social structure.

Lemma 2. *When $\mathcal{L}$ is a context free grammar and the social structure graph is a star, then MAPR($\mathcal{T}_{T\times n}, \mathcal{L}, f$) can be solved in time $O(n^2G^2T^3)$ for additive f.*

Theorem 3. *When $\mathcal{L}$ is a context free grammar and the social structure graph is a tree of bounded depth d, then MAPR($\mathcal{T}_{T\times n}, \mathcal{L}, f$) can be solved in time $O(n^{d+1}G^2T^3)$ for additive f.*

Another case of static teams with a known social structure can be made from a path, which is a special tree, but of variable depth.

Lemma 4. *When $\mathcal{L}$ is a context free grammar and the social structure graph is a path, then MAPR($\mathcal{T}_{T\times n}, \mathcal{L}, f$) can be solved in time $O(n^5G^2T^3)$ for additive f.*

Dynamic Teams

Despite the positive results of the previous section, there are many scenarios where the team structures among the observed agents can indeed change dynamically within the observation horizon. Tracking dynamic teams has received some attention in the past, initially as a part of broader environmental dynamism (Tambe 1996), but with greater focus more recently (Sukthankar and Sycara 2006; Avrahami-Zilberbrand and Kaminka 2007). However, the hardness of this problem has been unknown so far. In this section we show that MAPR is NP-complete when teams can be dynamic, even when the social structure of the observed agents is as simple as a path. This indicates, that for more complex and realistic social structures such as hierarchical/tree structures, the problem will remain NP-hard since a path is a special case of a tree. Note that the team sizes are variable here.

Theorem 5. *When $\mathcal{L}$ is a finite collection of matrices, and the social structure graph is a path, then MAPR($\mathcal{T}_{T\times n}, \mathcal{L}, f, k$) is NP-complete for the class of polynomially computable f.*

Proof: Since the social structure is a path graph, the agents/columns of the trace matrix can be (re)arranged such that the column-ordering matches the vertex ordering in the given path. Then every occurrence is *rectangular*, i.e., contiguous on both the time and the agent dimensions, since teams are only allowed on subpaths of the given social structure. Thus the problem of partitioning the trace reduces to that of rectangular partitioning of the trace matrix.

We reduce an arbitrary instance of the RTILE problem (Khanna, Muthukrishnan, and Paterson 1998) to a special instance of MAPR. See (Banerjee, Lyle, and Kraemer 2011) for the relevant details of RTILE. For the instance (B, p, u) of RTILE, we create a MAPR instance as follows. We create an $n \times n$ matrix of the same symbol for the trace $\mathcal{T}$, where B is of size $n \times n$. To create the plan library, $\mathcal{L}$, we form submatrices of the same symbol, of size $i \times j$, for all $i = 1 \ldots n, j = 1 \ldots n$. Thus any contiguous submatrix of $\mathcal{T}$ matches some element of $\mathcal{L}$, and hence is in fact an occurrence. More importantly, such a contiguous submatrix of $\mathcal{T}$, or an occurrence o, corresponds to a rectangular tile of the matrix B, and hence can be associated with the corresponding tile value, call it v_o. We assign value to each occurrence o as $v(o) = v_o/K$, where $K = \sum_{i,j} B[i,j]$. Finally, for the utility function f we choose the polynomially computable function

$$f(o_1, o_2, \ldots, o_z) = (1 - |p - z|).(1/(1 + \max_i v(o_i) - u/K))$$

and set $k = 1$. This instance of MAPR has a solution with value $\geq k$ iff RTILE(B, p, u) has a solution. The proof of inclusion in NP is similar to (Banerjee, Kraemer, and Lyle 2010). □

Although the above proof shows that there is some polynomial f for which this class of MAPR is hard, the chosen f has such a specific form that it could leave one wondering whether an additive f could still admit a polynomial time solution. We leave this avenue for future work.

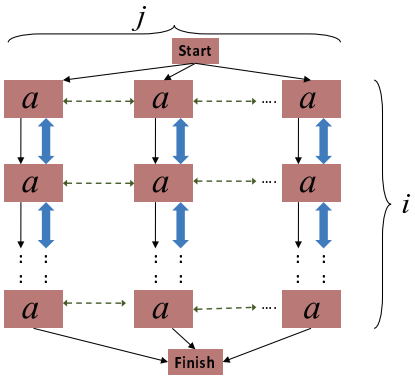


Figure 2: The plan graph corresponding to the $i \times j$ matrix plan. The two-headed dashed arrows represent concurrency constraints, the solid arrows represent ordering constraints, and the two-headed thick arrows represent role constraints.

The proof of Theorem 5 can be readily extended to cover plan libraries that are finite collections of plan graphs (as presented in illustration in Figure 1(c)). Such a plan library itself can engender an infinite collection of plan matrices, indicating that this problem is no easier than that addressed in Theorem 5. The proof proceeds the same way as Theorem 5, except that instead of the plan submatrix (using the

same symbol, a, say) of size $i \times j$, we create the plan graph shown in Figure 2.

Corollary 1. *When $\mathcal{L}$ is a finite collection of plan graphs, and the social structure graph is a path, then MAPR($\mathcal{T}_{T\times n}, \mathcal{L}, f, k$) is NP-complete for the class of polynomially computable f.*

Essentially, the representation of the plan library as a finite set of matrices becomes a special case of the plan graph representation, and possibly other (more useful) representations as well. In fact, Theorem 5 establishes a baseline such that this problem in conjunction with any practically interesting plan library (such as an HTN or a context free grammar) should be at least as hard as any NP-complete problem.

Interleaved Plan Execution

Interleaved execution of plans have been accommodated in activity recognition in the past (Chai and Yang 2005; Hu and Yang 2008), where an agent can interrupt a plan to serve another and resume it later. In a multi-agent system, this means that an agent can also be a part of another team in the interim, or leave a team and join another while the former team plan is still in progress. Although heuristic approaches have been proposed to address interleaving, there has been no formal investigation into the hardness of interleaved plan recognition even with a single agent. Below, we prove the hardness of this problem for the first time, by showing a reduction from X3C (exact cover by 3-sets), a known NP-complete problem (Garey and Johnson 1979).

Theorem 6. *When $\mathcal{L}$ is a finite collection of strings $\in \Sigma^*$ (i.e., the finite matrix library for a single agent) or plan graphs (as with Theorem 5 and Corollary 1), and the agent is allowed to execute plans in an interleaved manner, then MAPR($\mathcal{T}_{T\times 1}, \mathcal{L}, f, k$) is NP-complete where f is polynomially computable (and even additive).*

Proof: First we note that a certificate can be given as $\langle(\tau_1^1, \tau_2^1, \ldots, \pi_1), (\tau_1^2, \tau_2^2, \ldots, \pi_2)\ldots\rangle$, where $\pi_1, \pi_2 \ldots$ are the plans that the agent has executed, and τ_i^j is the time (in $[1, T]$) when it executed the ith step of the plan π_j. The certificate is clearly of linear size, and can be verified in polynomial time as in (Banerjee, Kraemer, and Lyle 2010).

Next, we polynomially reduce a general instance of X3C (with X and $C \subseteq 2^X$, s.t. $|X| = 3q$, and $c_i \in C \Rightarrow |c_i| = 3$) to a special instance of MAPR($\mathcal{T}_{T\times 1}, \mathcal{L}, f, k$) with interleaving as follows. We assume $|\Sigma| \geq 3$, and choose τ such that $\tau < |X|$ but $(|\Sigma| - 1)^\tau \geq |X|$. Then for each element $x_i \in X$ we create a string, $s(x_i)$ of length τ by sampling with replacement from $\Sigma \setminus \{\alpha\}$, such that $s(x_i) \neq s(x_j)$ if $x_i \neq x_j$. The selected symbol α is never used in this process, and is reserved for later. We order the elements of X and $c \in C$ in lexicographic order on the $s(x_i)$s. Then for each ordered $c_i \in C$ given by $c_i = \{x_i^1, x_i^2, x_i^3\}$, we create a string $\pi_i = s(x_i^1) \cdot \alpha \cdot s(x_i^2) \cdot \alpha \cdot s(x_i^3) \cdot \alpha$ of length $3(\tau + 1)$, where $\cdot$ is the string concatenation operator. We call the set of $|C|$ π_is the plan library $\mathcal{L}$. We choose the string $s(x_1) \cdot \alpha \cdot s(x_2) \cdot \alpha \cdot \ldots s(x_{|X|}) \cdot \alpha$ based on ordered X as the trace $\mathcal{T}$ of length $T = |X|(\tau + 1)$. Figure 3 shows an illustrative example of this reduction.

We claim that this setting of MAPR has a solution with q plans iff X3C has a cover of size q. To solve MAPR, we clearly need to cut $\mathcal{T}$ at various positions so that (possibly discontiguous) segments can be joined to reflect $\pi \in \mathcal{L}$. We call a cut beneath any α a *legal* cut, while a cut beneath any symbol in the trace string that comes from the set $\Sigma \setminus \{\alpha\}$ is called an *illegal* cut. It is easy to see that if all cuts in a MAPR solution are legal, then a solution to the X3C instance can be trivially constructed. Furthermore, if a solution to X3C exists, it can trivially produce a solution to this setting of MAPR, using legal cuts only. On the other hand, if a solution to MAPR could incorporate illegal cuts, then this one-to-one correspondence between the solutions of MAPR and those of X3C would break down, but Lemma 7 shows that no cut in a MAPR solution can ever be illegal. □

Lemma 7. *A solution to the above instance of MAPR must only incorporate legal cuts.*

Proof (by contradiction): Suppose there is one or more illegal cut in the solution. Consider the bottom-most illegal cut – say the kth cut from the top– and the segment ending at this cut. Call this the *open* segment. Since the open segment does not end in α, it must be appended by one or more segments that lie between the $(k + 1)$th and the subsequent cuts, to produce a complete plan. However, *all* subsequent cuts (if any) are *legal* by assumption. That is, all intervening segments in these legal cuts are of lengths that are multiples of $\tau + 1$. Observe that no number of such segments can complete the open segment, because the number of symbols between the last α (or the beginning) of the open segment and the next α in the completed (i.e., appended) plan must necessarily exceed τ. This violates every plan in the library by construction, and therefore we do not have a solution– a contradiction. □

Contrasting this result with the polynomial solvability of MAPR without interleaving and with a single agent (Banerjee, Kraemer, and Lyle 2010) reveals the impact of interleaved plan execution on the plan recognition problem. Since MAPR with multiple agents and interleaving cannot be any easier, this also offers evidence of the hardness of the latter.

Conclusions

We have presented two important extensions to our recent formalization of MAPR, to accommodate compact multi-agent plan libraries and incomplete plans. We have studied several special cases of MAPR with static teams, bounded team sizes, and known social structures, and shown how these can be solved in polynomial time. Unfortunately, with dynamic teams and social structure even as simple as a path, MAPR turns out to be NP-complete. Moreover, when activity interleaving is allowed, even the single agent problem turns out to be NP-complete, implying the hardness of multi-agent interleaved plan recognition. From the point of adversariality, a single agent can render the recognizer's problem from P to NP-complete by interleaving activities, but with multiple (and a variable number of) agents, the problem is NP complete with or without activity interleaving. In other words, interleaving is an effective adversarial tool for a sin-

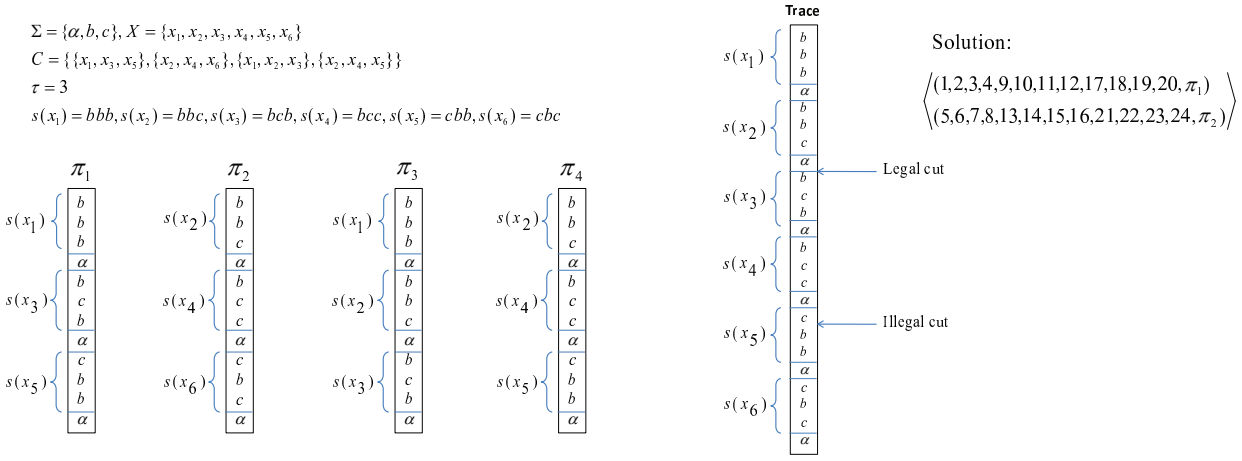


Figure 3: Illustration of the reduction from an instance of X3C to MAPR with interleaving for a single agent. Part (a) shows the setup and the plan library consisting of π_1–π_4. Part (b) shows the trace and the solution.

gle (observed) agent against a recognizer, but it is not as effective for multiple (observed) agents.

Acknowledgments

We are thankful to the anonymous reviewers, and also to Gal Kaminka and Gita Sukthankar, for valuable suggestions and comments. This work was supported in part by a start-up grant from the University of Southern Mississippi.

References

Avrahami-Zilberbrand, D., and Kaminka, G. A. 2007. Towards dynamic tracking of multi-agent teams: An initial report. In *Proceedings of the AAAI Workshop on Plan, Activity and Intent Recognition (PAIR-07)*.

Banerjee, B.; Kraemer, L.; and Lyle, J. 2010. Multi-agent plan recognition: Formalization and algorithms. In *Proceedings of AAAI-10*, 1059–1064.

Banerjee, B.; Lyle, J.; and Kraemer, L. 2011. New algorithms and hardness results for multi-agent plan recognition. Technical report. Available at `http://www.cs.usm.edu/~banerjee/tr/longer.pdf`.

Chai, X., and Yang, Q. 2005. Multiple-goal recognition from low-level signals. In *Proceedings of the AAAI*, 3–8.

Erol, K.; Hendler, J.; and Nau, D. S. 1994. HTN planning: Complexity and expressivity. In *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94)*, 1123–1128. AAAI Press.

Garey, M. R., and Johnson, D. S. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. San Francisco, CA: W.H. Freeman and Co.

Ghallab, M.; Nau, D.; and Traverso, P. 2004. *Automated Planning: Theory and Practice*. Morgan Kaufmann Publishers.

Hu, D. H., and Yang, Q. 2008. Cigar: Concurrent and interleaving goal and activity recognition. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence*, 1363–1368.

Kaminka, G.; Pynadath, D.; and Tambe, M. 2002. Monitoring teams by overhearing: A multi-agent plan recognition approach. *Journal of Artificial Intelligence Research* 17.

Kautz, H. A., and Allen, J. F. 1986. Generalized plan recognition. In *Proc. AAAI*.

Khanna, S.; Muthukrishnan, S.; and Paterson, M. 1998. On approximating rectangle tiling and packing. In *Proceedings of the ninth annual ACM-SIAM symposium on Discrete algorithms*, SODA '98, 384–393. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics.

Sadilek, A., and Kautz, H. 2010. Recognizing multi-agent activities from gps data. In *Proceedings of AAAI-10*, 1134–1139.

Sukthankar, G., and Sycara, K. 2006. Simultaneous team assignment and behavior recognition from spatio-temporal agent traces. In *Proceedings of AAAI conference*.

Sukthankar, G., and Sycara, K. 2008. Hypothesis pruning and ranking for large plan recognition problems. In *Proc. of AAAI*.

Tambe, M. 1996. Tracking dynamic team activity. In *Proc. of AAAI*.

Vail, D. L., and Veloso, M. M. 2008. Feature selection for activity recognition in multi-robot domains. In *AAAI'08: Proceedings of the 23rd national conference on Artificial intelligence*, 1415–1420. AAAI Press.

Vilain, M. 1990. Getting serious about parsing plans: a grammatical analysis of plan recognition. In *Proc. of AAAI-90*.

Accurately Determining Intermediate and Terminal Plan States Using Bayesian Goal Recognition

David Pattison and **Derek Long**
Department of Computer and Information Science
University of Strathclyde, Glasgow G1 1XH, UK
{david.pattison, derek.long}@cis.strath.ac.uk

Abstract

Goal Recognition concerns the problem of determining an agent's final goal, deduced from the plan they are currently executing (and subsequently being observed). The set of possible goals or plans to be considered are commonly stored in a *library*, which is then used to propose possible candidate goals for the agent's behaviour.

Previously, we presented AUTOGRAPH – a system which removed the need for a goal or plan library, thus making any problem solvable without the need to construct such a structure. In this paper, we discuss IGRAPH, which improves upon its predecessor by utilising Bayesian inference to determine both *terminal* and *intermediate* goals/states which the agent being observed is likely to pass through.

1 Introduction

Goal Recognition (GR) can be considered a sub-problem of Plan Recognition (PR) where only the *terminal* goal is required and the plan used to achieve this is somewhat irrelevant. Traditionally, both of these fields have made use of *libraries* (Kautz and Allen 1986; Goldman, Geib, and Miller 1999) which contain known, valid plans or goals and the plans used to achieve them, commonly represented as Hierarchical Task Networks (HTNs) (Nau, Ghallab, and Traverso 2004). Construction of these libraries by domain-expert is a time-consuming task, while automatic generation suffers the risk of being incomplete or containing irrelevant and invalid entries.

In our previous work with AUTOGRAPH (Pattison and Long 2010), we removed the need for a plan or goal library by representing the problem as a Planning task in which observations reflected movement through the *state space* of the associated domain. After each observation O a hypothesis was produced which represented a belief in the agent's final goal. Perhaps expectedly, the accuracy of these hypotheses was often directly correlated to the number of plan steps observed so far. That is to say, more accurate hypotheses were produced towards the end of the plan, while earlier hypotheses often lacked many of the final goals as no previous observation had indicated they were a part of the true goal. Furthermore, AUTOGRAPH assumed a near-optimal (or even optimal) plan was being observed – something which is rarely true in real-life.

This paper presents IGRAPH (Intermediate Goal Recognition with A Planning Heuristic), a recognition engine which tackles the problem of accurately determining an agent's *intermediate* goal states – that is, states which the agent is expected to pass through **before** the end of their plan. To do this we adopt a Bayesian approach to reasoning over which goals are most likely at future timestep t, and furthermore provide a *relaxed estimate* of remaining plan length.

2 Motivation for Intermediate Goal Recognition

Tackling Goal Recognition without any form of library is a far harder task than if one were available. With such a library present the number of possible goals or plans being pursued is trivial in comparison to having to consider the entire state-space and all possible plans. However, for most real-world problems the benefits of a library-based recognition system are eclipsed by the scarcity of useful candidate plans and goals.

To motivate the need for non-library based recognition, let us consider a typical city shown in Figure 1a which is made up of many buildings, streets and services, with people in the city able to move between and interact with these. Given that we are to observe the movements of just a single person around this city, the number of possible plans which a library would have to contain for even a fixed starting location is intractable. However, if we consider each achievable fact to be a single, independent goal, then enumeration of all destinations or tasks becomes possible.

Now let us assume we begin observing an agent who starts at location A, then proceeds to walk to locations E and G before stopping at F (as shown in Figure 1b). If we assume the agent only has a single goal which is to move to another location, then as the plan progresses we can eliminate destinations which become harder to reach after each observation. For example, after the agent has moved from A to E, it can be deduced that they are probably not trying to reach B, D or H, as the route they have already chosen would mean a longer plan than had they moved directly towards one of these after starting at A.

Once the agent has reached E, we can reason that they are heading to C, G or F. Now suppose we wish to interact with

the agent **before** they achieve any of these goals. If we can deduce that they must pass through at tollbooth at X first, then we can plan to stop, hinder or help them in achieving their final destination goal.

However, the assumption in the above example that there will only ever be a single goal is both restricting and unrealistic. It is far more likely that the agent would have several goals. For instance, they may stop at location E to buy some goods, visit a relative at location G, then continue to location B. Their plan may be only 1 step long, or it may run to hundreds of actions. The ability to dynamically generate valid goal-conjunctions from otherwise unrelated goals is central to IGRAPH. This is in contrast to most previous work which assumes plans achieve a known, fixed conjunction or a single-goal (Mott, Lee, and Lester 2006; Lesh and Etzioni 1996).

Expanding this example to that of a real-world formulation, GR is often associated with monitoring and tracking (Huntemann et al. 2008; Geib and Goldman 2001) or in military and video game simulations (Schadd, Bakkes, and Spronck 2007; Kabanza et al. 2010; Albrecht, Zukerman, and Nicholson 1998; Cheng and Thawonmas 2004). While determining the agent's **final** goal is the crux of GR, in all of these cases it would be beneficial to know the **intermediate** states or goals which the agent being observed is most likely to pass through on their way to achieving their final goals. Having this knowledge could be used to prevent the agent from achieving specific undesirable goals, or as an aid to plan deduction by using these intermediate states in a similar manner to *landmarks* (Porteous, Sebastia, and Hoffmann 2001; Richter, Helmert, and Westphal 2008). Conversely, in a co-operative domain with two or more agents in which communication has been lost but execution is continuing, the ability to pre-empt another agent's intermediate goals could allow for co-operative tasks to be executed with minimal interruption and aid plan repair.

Many applications of GR relate strongly to *adversarial* recognition, wherein agents actively try to block their plans or goals from being recognised by the observer (Geib and Goldman 2001). Yet the majority of this work is related to simply the *detection* of adversarial behaviour, not the prevention of the outcome. It is also true that much of this has been directed towards plan recognition as opposed to goal recognition. Conceptually, this means that the problem of detecting intermediate goals is a moot point, as any plan hypothesis will allow intermediate states to be computed without probabilistic reasoning being required. This is the case in Blaylock and Allen's work (2006), in which they experiment with the use of HTNs containing a high-level goal at the root, which decomposes into ordered subgoals (which may also decompose themselves). Intermediate and high level goals are inferred by noting that a subgoal at layer n has been achieved after an observation, and producing a *goal chain* – a path from the subgoal to the root.

Recently, the field of *interactive entertainment* has produced several works on recognition, based on Bayesian inference (Charniak and Goldman 1993) where the intention is to determine the goals of an agent (commonly the player), such that a personal story can be crafted around their experience. In (Albrecht, Zukerman, and Nicholson 1998), Dynamic Belief Networks were used to generate predictions of a player's next action and current quest in the context of a text-based virtual-reality game, which were trained using recordings of observed plans and actions. Mott *et al* (2006) offer another Bayesian approach to GR in the detection of goals in an interactive narrative environment. Their results show that goals can be incrementally converged upon as more evidence is provided, but that only single-goals are considered. More recently, (Kabanza et al. 2010) have presented HICOR, a PR system for detecting an opponent's intentions in a real-time strategy game. While HICOR can present multiple, concurrent plans as a single hypothesis, testing has been performed using only mutually-exclusive single goals (although detecting multiple goals is possible).

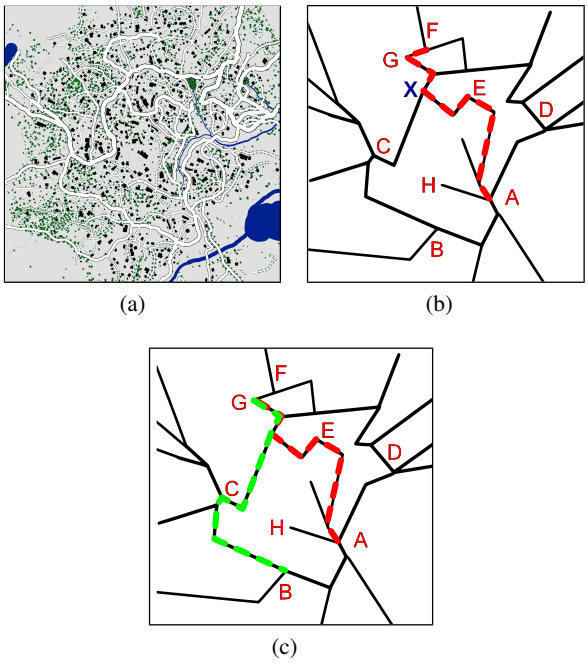


Figure 1: A small city map and several example plans displayed on a heavily-simplified version.

Ramírez and Geffner (2010) also present a Planning-based model of GR similar to IGRAPH, which computes goal probability based on heuristic estimates after each observation. Probability likelihoods are based on *cost differences*, which represent the cost of achieving $P(G|O)$ versus $P(G|\neg O)$. While their work allows for multiple plans achieving multiple goals, it is evaluated using domains which comprise of only a few hundred actions and goals, while IGRAPH can handle thousands or even tens-of-thousands of goals.

3 Problem Definition

We begin by defining our representation of the problem. As in the original AUTOGRAPH we use a propositional Planning model similar to a STRIPS encoding (Fikes and Nilsson 1971), which we derive from a PDDL domain and problem

definition (McDermott et al. 1998; Fox and Long 2003).

Definition 1. *Goal Recognition Problem Base*
A goal recognition problem base is a triple $\langle F, A, I\rangle$, where F is a set of primitive (propositional) facts, A is a set of actions and $I \subseteq F$ is the initial state for the problem. Each action $a \in A$ is a triple $\langle a_{pre}, a_{add}, a_{del}\rangle$, where $a_{pre}, a_{add}, a_{del} \subseteq F$ are the preconditions, add effects and delete effects of a, respectively.

We also require a *Goal Recognition Problem* representing the plan being observed. Unlike AUTOGRAPH we do not assume that the plan being observed is optimal, near-optimal, or even rational. However, we do retain the assumptions that the plan is fully-observable and totally-ordered, but that we do not know its length. As we will discuss shortly, this final assumption is key to the operation of IGRAPH.

Definition 2. *Goal Recognition Problem*
A goal recognition problem is a triple, $\langle \mathcal{G}, \mathbb{H}_I, P\rangle$, where $\mathcal{G}$ is a goal recognition problem base, $\mathbb{H}_I$ is an initial probability distribution over the hypothesis space $\mathbb{H}$ and $P = \langle o_1, ..., o_n\rangle$ is the sequence of plan actions observed one-by-one during the problem.

Enumeration of $\mathbb{H}$ is intractable for all but the most trivial of problems, as it is a superset of the state-space. Thus, we define a *relaxed hypothesis space*, $\mathcal{H}$, which contains each individual *reachable fact*, f, and note sets of known mutually-exclusive facts[1] in order to keep the probability distribution over $f \cup mutex(f)$ normalised. However, while we only enumerate individual goals, we still allow for multiple facts to be considered as the goal of plan P, i.e. goal G_1 can be achieved, then goals G_2 and G_3 achieved in later observations. The mechanics of this last point are discussed in Section 4.

Before recognition begins, we may assign a *uniform probability distribution* over all mutually-exclusive facts, which become the *prior probabilities* for Bayesian inference. Alternatively, we may assign a *weighted probability distribution* using the same domain analysis techniques used in AUTOGRAPH (Pattison and Long 2010). Once all goals have an initial probability we compute their *heuristic estimate* $h(G)$ – an approximate measure of the number of actions required to achieve G.

As the agent executes actions, they move through the state-space and subsequently will move closer to achieving certain facts/goals and further away from others. This movement can be used as an indication of which subset of $\mathbb{H}$ is being pursued. We update the heuristic estimate for all goals $G \in \mathcal{H}$ after each observation. These new estimates are then used to compute the *posterior probability* $P(G|O)$.

4 Heuristic Estimates as Bayesian Likelihoods

In AUTOGRAPH we also made use of heuristic estimation to determine the probability of a fact f being the true goal. However, due to the assumption of having an optimal/near-optimal plan, if $h(f)$ increased after an observation its probability of being a goal was reduced to zero regardless of its previous value. This behaviour is also in line with use of an optimal heuristic, something which does not exist, thus hypothesis quality could be affected by inaccurate estimates.

For IGRAPH we have adopted an *interpolated* Bayesian approach to probability updates which has been inspired by work in Information Retrieval (IR) (Zhai and Lafferty 2004). We use observation O to update the probability of each goal $P(G|O)$ by computing the *likelihood function* $P(O|G)$ using $h(G)$. This determines the probability of O being relevant if G is assumed to be the true goal. By using interpolated smoothing to compute $P(G|O)$ we remove the ability for $(O|G)$ and thus $P(G)$ to equal zero. This shift to evidence-based probability updates means that goals can no longer be completely eliminated from the set of hypothesis *goal candidates*. Furthermore, it allows the agent to revisit sections of the search-space after having achieved a goal in another section.

We now define the amount of work expended on a goal G after an observation has been processed. Given observation O and a set of mutually-exclusive goals $\bar{G}$, the proportion of work which has been expended in moving towards achieving G is shown in Equation 1, where λ is a smoothing factor $\lambda \in [0:1]$ and $\bar{G}^{nearer}$ is a set of mutually-exclusive goals (including G) whose heuristic estimate has lowered after A_t has been observed, or whose estimate has been zero for at least the past two observations.

$$W(G|O) = \begin{cases} \frac{1}{|\bar{G}^{nearer}|} & \text{if } h_t(G) < h_{t-1}(G), \\ \frac{1}{|\bar{G}^{nearer}|} & \text{if } h_t(G) = h_{t-1}(G) = 0, \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

In Section 2 we discussed the requirement of being able to have **any** valid conjunction of goals $G \in \mathcal{H}$ in a hypothesis. We implement this by introducing a relaxation in the assignment of $W(G|O)$. By providing goals which have remained true over timesteps t and $t-1$ with a *bonus* of 1, we encourage goals which have been achieved to remain valid goal candidates. Consider the simple ZENOTRAVEL problem shown in Figure 2, in which the plan being observed will pick up passenger 1 from city 2, drop them off at city 1, fly to pick up passenger 1 in city 3, then return to city 1. After observing action `7:[fly city2 city1]`, the goal `(at city1 passengerA)` will have been true for 3 timesteps and no mutually-exclusive facts will have become heuristically closer. However, once action 7 has been observed, the heuristic estimate to `(in plane passenger1)` starts to reduce again. Without the bonus being applied to facts which have remained true in the intermediate timesteps, the probability of these goals reduces (as $P(O|G)$ is low), while the probability of others can increase.

This example also highlights another assumption made by IGRAPH – namely that once achieved, an agent will strive to keep the goal true if possible. We refer to this as the *stability*

[1] IGRAPH does not assume that *all* mutually-exclusive facts for a given domain are known, although not knowing this may affect the accuracy of the final probability distribution.

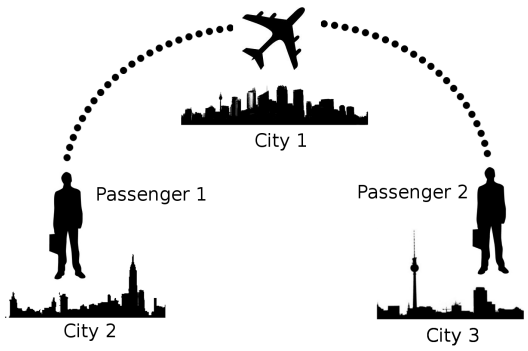


Figure 2: A small ZENOTRAVEL problem which highlights the need for a bonus being applied to goals which have remained true over $n > 0$ consecutive timesteps.

of the goal (see Equation 2). The stability of a goal $S(G)$ indicates how often it has been achieved, then *unachieved* in a later observation, with the first achievement being denoted as G_t^{true} and $\sum G_i^{true}$ the total number of timesteps G has been true since first achievement. In the above example, `(at city1 plane1)` is unachieved and re-achieved several times, giving it a low stability relative to other goals such as `(at city1 passengerA)`.

We note that the use of bonus scores is not without its risks. Consider a variable with 2 mutually-exclusive transitions $\{F_1, F_2\}$, and that F_1 is true initially. If F_1 can transition to F_2 at any time during the observation of the first k plan steps (i.e. $h(F_2) = 1$), then the probability of F_2 will never increase, while $P(O|F_1)$ will receive a bonus after every observation. If after k observations, the final plan action is observed which transitions F_1 to F_2, the probability increment for F_2 will be so small as to make no difference to hypothesis generation – more evidence would be needed to rule out F_1 as the true goal.

While this behaviour is perceivable for some problems, in general the above example would be unlikely. It is also arguable that if F_1 holds true for a long time before transitioning on the last known observation O_k, that it is probable F_1 *was* the goal for the plan observed until O_k, and that now a different plan with different goals is being pursued – or at the very least F_1 was an intermediate goal.

We incorporate the stability of a goal into the Bayesian likelihood function when computing the new posterior probability for each goal.

$$S_t(G) = \begin{cases} 1 & \text{if } G \text{ unachieved in } P, \\ \dfrac{|Obs| - G_t^{true}}{\sum G_i^{true}} & \text{otherwise} \end{cases} \quad (2)$$

$$P(O|G) = \lambda * W(G|O) * S(G) + (1 - \lambda) * \frac{1}{|\bar{G}|} \quad (3)$$

$$P(G|O) = \frac{P(G)P(O|G)}{\sum P(G_i)P(O|G_i)} \quad \forall G_i \in \bar{G} \quad (4)$$

Given a goal G and the set of goals which are mutually-exclusive $mutex(G)$, the *interpolated likelihood function* shown in Equation 3 defines the probability of O being relevant to the achievement of G with respect to all other goals $G_i \in \bar{G}$, where $\bar{G} = G \cup mutex(G)$. The smoothing factor λ prevents any goal from receiving a value of zero for $P(O|G)$, with low values causing the probability distribution over $\mathcal{H}$ to be more evenly spread amongst mutually-exclusive facts.

As as special case, if a goal has no mutexes Equation 3 will not suffice as $P(O|G)$ for any stable goal which has moved closer will be 1. For this we use *laplace smoothing*, another IR scoring technique shown in Equation 5, where $\mu \in \mathbb{Z}^+$ and $|O_{helpful}|$ is number of observations which have lowered the original estimate $h(G)$, or maintained it at zero over $n \geq 2$ steps.

$$W(G|O) = \frac{|O_{helpful}| + \mu}{|O| + \mu} \qquad \text{iff } |mutex(g)| = 0 \quad (5)$$

We now describe how this Bayesian approach to GR is used to generate both *intermediate* and *terminal* hypotheses.

5 Hypotheses as Intermediate States

Once the probability of every goal has been updated after each observation, we can attempt to estimate the number of remaining steps within the plan $\varepsilon \in \mathbb{Z}^+$. This is computed by generating an *immediate goal hypothesis* H_I, which is simply the set of mutually-exclusive facts that have the highest probabilities within $\mathcal{H}$, then heuristically estimating the number of steps required to achieve the hypothesis, $\varepsilon = h(H_i)$.

After ε has been computed, a *bounded hypothesis* can be generated for the next n steps, which is equivalent to the set of facts that are expected to be true at time $t + n$ (or which of the facts in the current state are the goal if $n = 0$). If the hypothesis contains a value from every set of mutually-exclusive facts, it is a *bounded intermediate state*, while if $n = \varepsilon$ it is a *terminal hypothesis*.

Definition 3. *Bounded Goal Hypothesis*
*A **bounded goal hypothesis** H_t^n is a set of non-mutually-exclusive facts $\{G_1, G_2 ... G_k\}$, where $H_t^n \in \mathbb{H}$ produced at time t on the the rationale that H will be true at time $t + n$, and that $0 \leq n \leq \varepsilon$.*

Facts which are a member of the relaxed-goal-space $\mathcal{H}$ are selected for a bounded hypothesis based on the probability of an action A which achieves them being observed in the next n steps (see Equations 6 and 7). As an actions preconditions A_{pre} must all be satisfied before it can be applied and thus its effects added, the probability of these being an intermediate goal must also be considered.

$$P^n(A) = \begin{cases} 0 & \text{if } h(A_{pre}) > n, \\ \max P(f) \;\; \forall f \in A_{add} & \text{otherwise} \end{cases} \quad (6)$$

$$P^n(G) = \max P^n(A) \;\; \forall A \in achievers(G) \quad (7)$$

6 Evaluation

IGRAPH has been tested on the propositional versions of the DRIVERLOG, ZENOTRAVEL and ROVERS domains taken from the 3rd International Planning Competition along with their best-known-plan solutions (Long and Fox 2003). The FF heuristic (Hoffmann and Nebel 2001) has been used for evaluation, although any heuristic would suffice. A smoothing constant of $\lambda = 0.8$ was used for Bayesian updates, while $\mu = 1$ was used in the computation of $W(G|O)$ if $|mutex(G)| = 0$.

Tests were conducted in Ubuntu 9.10 on a quad core 2.8GHz Intel i5 with 4GB of RAM using the latest Java Virtual Machine (1.6.0_20), and were given as much time as necessary to complete each stage of the recognition process.

6.1 Intermediate Bounded Hypotheses

As in AUTOGRAPH, intermediate hypotheses produced by IGRAPH have been evaluated using *precision and recall* (P/R). However, while previously the P/R of a hypothesis was compared with the agent's true final goal, here they are compared against the **state** encountered at time $t + n$.

The results of all intermediate hypotheses over each domain tested are shown in Figure 3. P/R results have been rounded to the nearest two decimal places in order to group together results for easier reading. The radius of a circle indicates the number of results in each grouping.

In the case of DRIVERLOG, clustering results are largely grouped above P/R = 0.5/0.5 indicating that the majority of *intermediate hypotheses* are reasonably accurate for their target bound. Results for ROVERS are primarily distributed across R = 0.45, while ZENOTRAVEL also displays strong clustering around P/R = 0.5/0.5.

Figure 4 displays intermediate clustering results in relation to their bound n. For example, column 1 relates to the accuracy of all hypotheses with $n = 1$ (hypotheses which are expected to be true after 1 further observation). These results reveal two details: that the majority of results are above 50% accurate, but also that there is a large number of P/R scores equal to (0.5/0.5) and that the estimation of remaining plan steps is often only a few steps from the current state (with $\varepsilon = 7$ being the highest observed). With regard to the former, the exact reason for the clustering around (0.5/0.5) is currently unknown as these results are spread across all three test domains. The latter observation can be explained by a combination of the accuracy of the heuristic used for estimation and the assumption that we have no knowledge of when the plan will terminate.

6.2 Short Lookahead Estimation

The nature of the FF heuristic means that estimates to goals which are far from the current state will often be much lower than their true distance, while goals that are closer will have a more accurate estimate. In fact, it is not uncommon for the estimate to distant goals to remain the same (or even increase) over multiple observations, despite the fact they are actually becoming closer. This apparent lack-of-progress towards a goal G alone is enough to eliminate it as a candidate for hypotheses, as its probability will likely be low. This

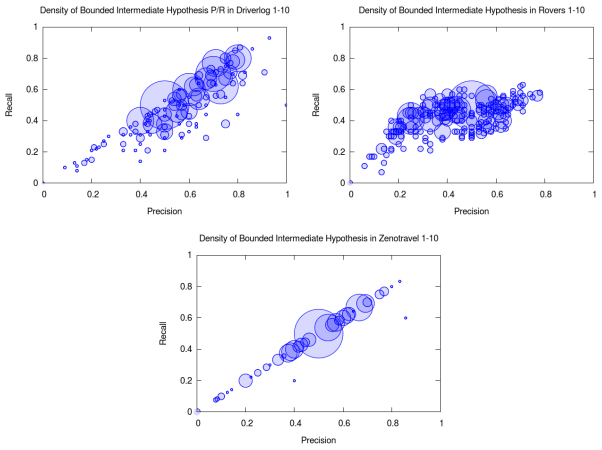


Figure 3: Density of P/R rounded to 2 d.p. over problems 1-15 and over all intermediate hypotheses for DRIVERLOG, ROVERS and ZENOTRAVEL. Circle radius is a reflection of the number of results at a specific P/R value.

is further compounded by the chance of multiple mutually-exclusive facts becoming closer after each observation.

Poor heuristic estimates for distant goals can also explain low values for ε, as only a consistent decrease in $h(G)$ will ensure they are considered as part of the *immediate hypothesis* from which ε is deduced.

6.3 Terminal Hypotheses

As stated previously, while intermediate goal hypotheses are undoubtedly useful, the ultimate task in GR is to determine the agent's final goal. Thus, we produce a single *terminal hypothesis* after each observation which is equivalent to the *immediate hypothesis* produced to detect ε. The P/R results of these hypotheses when computed at various stages of plan observation is displayed in Table 1.

The results show a high average value for *recall* over all problems, while *precision* is often lower. However, as previously stated, while *bounded hypotheses* are tested against full states, *terminal hypotheses* are tested against the true goal only. As both hypothesis types are generated from the same algorithm, the precision for terminal hypotheses will often be much lower than the bounded equivalent.

In both DRIVERLOG and ZENOTRAVEL, P/R results show a faster *convergence* upon the correct goal than previously displayed in AUTOGRAPH. ROVERS results display both the lowest average precision yet perfect recall. The nature of typical goals in a ROVERS problem is what causes these static results. Most often the goal is to achieve communication of a rock sample or photograph, but that this can be performed in several ways. In the case of an image, once obtained it can be transmitted at a low or high resolution, or in colour, but that crucially any combination of these is possible. Thus as an agent progresses through a plan wherein they move to a sample and photograph it, the probability of communicating the image in all forms increases at the same rate, as each type of `communicated` goal is non-mutex. This means that hypotheses will include all types

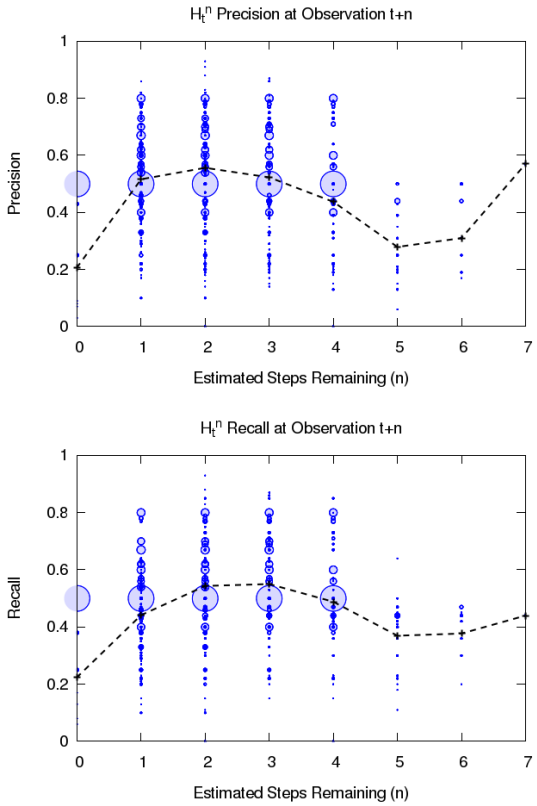


Figure 4: Density and averages for P/R of all hypotheses for all tests over all domains, when compared against the state encountered after n further observations.

of `communicated` goal, despite there often only being a single one required.

Finally, we note that average recall results for IGRAPH are considerably higher than those of AUTOGRAPH, with an average of 59% recall without any observations even being required. However, while recall has increased over all domains, precision has lowered for $|P| = 25 - 100\%$. This is caused by *terminal hypotheses* being produced in the same manner as *bounded hypotheses*, wherein the hypothesis is closer to a full state specification rather than a subset of goals. In AUTOGRAPH, many more restricting assumptions were made about the set of goals considered as candidates, which ultimately allowed for more concise hypotheses.

Domain	**|P| = 0%**	**|P| = 25%**	**|P| = 50%**	**|P| = 75%**	**|P| = 100%**
Driverlog	0.22/0.3	0.33/0.45	0.46/0.6	0.55/0.69	0.66/0.84
Rovers	0.28/1	0.28/1	0.28/1	0.28/1	0.32/1
Zenotravel	0.28/0.46	0.23/0.39	0.25/0.43	0.36/0.63	0.4/0.68
IGRAPH Avg.	0.26/0.59	0.28/0.61	0.33/0.68	0.4/0.77	0.46/0.84
AUTOGRAPH Avg.	0.02/0.02	0.45/0.12	0.64/0.27	0.76/0.48	0.88/0.73

Table 1: A compilation of averaged P/R scores for all *terminal hypotheses* over all domains tested at 0%, 25%, 50%, 75% and 100% plan completion.

7 Discussion

We have presented IGRAPH, an extension of AUTOGRAPH which attempts to move the state-of-the-art forward in goal recognition by tackling the problem of recognising multiple unrelated goals and estimation of intermediate plan goals/states. This is done without the need of a plan or goal library, making it applicable in a wide range of situations with only minimal prior effort.

By generating both accurate intermediate and terminal hypotheses, we have laid the groundwork for expanding into non-library-based *plan recognition*. As each bounded hypothesis is essentially a stepping-stone towards the final goal state, the ability to determine action selection becomes a much simpler task. Indeed, initial experiments have shown that IGRAPH can produce highly accurate *next-action prediction*. By chaining these predictions together we may be able to derive a *predicted plan* without resorting to fully-blown planning.

References

[Albrecht, Zukerman, and Nicholson 1998] Albrecht, D. W.; Zukerman, I.; and Nicholson, A. E. 1998. Bayesian models for keyhole plan recognition in an adventure game. *User Modeling and User-Adapted Interaction* 8(1-2):5–47.

[Blaylock and Allen 2006] Blaylock, N., and Allen, J. F. 2006. Fast hierarchical goal schema recognition. In *Proc. Nat. Conf. on AI (AAAI)*, 796–801.

[Charniak and Goldman 1993] Charniak, E., and Goldman, R. P. 1993. A Bayesian model of plan recognition. *J. AI Res.* 64(1):53–79.

[Cheng and Thawonmas 2004] Cheng, D. C., and Thawonmas, R. 2004. Case-based plan recognition for real-time strategy games. In *Proceedings of the Game-On International Conference*, 36–40.

[Fikes and Nilsson 1971] Fikes, R., and Nilsson, N. J. 1971. STRIPS: A new approach to the application of theorem proving to problem solving. In *Proc. Int. Joint Conf. on AI*, 608–620.

[Fox and Long 2003] Fox, M., and Long, D. 2003. PDDL2.1: An extension to PDDL for expressing temporal planning domains. *J. AI Res.* 20:61–124.

[Geib and Goldman 2001] Geib, C. W., and Goldman, R. P. 2001. Plan recognition in intrusion detection systems.

[Goldman, Geib, and Miller 1999] Goldman, R. P.; Geib, C. W.; and Miller, C. A. 1999. A new model of plan recognition. *J. AI Res.* 64:53–79.

[Hoffmann and Nebel 2001] Hoffmann, J., and Nebel, B. 2001. The FF planning system: Fast plan generation through heuristic search. *J. AI Res.* 14:253–302.

[Huntemann et al. 2008] Huntemann, A.; Demeester, E.; van Brussel, H.; and Nuttin, M. 2008. Can Bayes help disabled users? A Bayesian approach to plan recognition and shared control for steering an electrical wheelchair. In *Proc. ACM/IEEE Human-Robot Interaction Conf.*, 67–70.

[Kabanza et al. 2010] Kabanza, F.; Bellefeuille, P.; Bisson, F.; Benaskeur, A. R.; and Irandoust, H. 2010. Opponent behaviour recognition for real-time strategy games.

[Kautz and Allen 1986] Kautz, H. A., and Allen, J. F. 1986. Generalized plan recognition. In *Proc. of AAAI-86*, 32–37.

[Lesh and Etzioni 1996] Lesh, N., and Etzioni, O. 1996. Scaling up goal recognition. In *KR*, 244–255.

[Long and Fox 2003] Long, D., and Fox, M. 2003. The 3rd international planning competition: Results and analysis. *J. AI Res.* 20:1–59.

[McDermott et al. 1998] McDermott, D.; Ghallab, M.; Howe, A.; Knoblock, C.; Ram, A.; Veloso, M.; Weld, D.; and Wilkins, D. 1998. PDDL - the planning domain definition language. Technical report, Yale Center for Computational Vision and Control.

[Mott, Lee, and Lester 2006] Mott, B.; Lee, S.; and Lester, J. 2006. Probabilistic goal recognition in interactive narrative environments. In *Proc. Nat. Conf. on AI (AAAI)*, 187–192.

[Nau, Ghallab, and Traverso 2004] Nau, D.; Ghallab, M.; and Traverso, P. 2004. *Automated Planning: Theory & Practice*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

[Pattison and Long 2010] Pattison, D., and Long, D. 2010. Domain Independent Goal Recognition. In *STAIRS 2010: Proceedings of the Fifth Starting AI Researchers' Symposium*, volume 222, 238 – 250. IOS Press.

[Porteous, Sebastia, and Hoffmann 2001] Porteous, J.; Sebastia, L.; and Hoffmann, J. 2001. On the extraction, ordering, and usage of landmarks in planning. In *Proc, European Conference on Artificial Intelligence*, 37–48.

[Ramírez and Geffner 2010] Ramírez, M., and Geffner, H. 2010. Probabilistic plan recognition using off-the-shelf classical planners. In *AAAI*.

[Richter, Helmert, and Westphal 2008] Richter, S.; Helmert, M.; and Westphal, M. 2008. Landmarks revisited. In *AAAI*, 975–982.

[Schadd, Bakkes, and Spronck 2007] Schadd, F.; Bakkes, S.; and Spronck, P. 2007. Opponent modeling in real-time strategy games. In *8th International Conference on Intelligent Games and Simulation (GAME-ON 2007)*, 61–68.

[Zhai and Lafferty 2004] Zhai, C., and Lafferty, J. 2004. A study of smoothing methods for language models applied to information retrieval. *ACM Trans. Inf. Syst.* 22:179–214.

Modeling the Human Operator's Cognitive Process to Enable Assistant System Decisions

Ruben Strenzke & Axel Schulte
Universität der Bundeswehr München
Department of Aerospace Engineering, Institute of Flight Systems
Werner-Heisenberg-Weg 39, 85577 Neubiberg, GERMANY
{ruben.strenzke, axel.schulte}@unibw.de

Abstract

Human operators in human-machine systems can be supported by assistant systems in order to avoid and resolve critical workload peaks. The decisions of such an assistant system should at best be based on the current and anticipated situation (e.g. mission progress) as well as on the current and anticipated cognitive state of the operator, which includes his/her beliefs, goals, plan, intended action, interaction with the environment, and subjective workload. The more of this can be assessed, the easier and the earlier human errors can be recognized and corrected by assistant system initiative. Multiple approaches to enable an assistant system to correctly decide whether, when, and in which way to take initiative are currently under research at the Universität der Bundeswehr München, e.g. mixed-initiative mission planning, which includes assuming the human operator's plan and estimation of operator workload by means of human operator behavior models. We here present our position in favor of an overarching framework for modeling a human operator, which is based on our Cognitive Process model.

Introduction

The Universität der Bundeswehr München (UBM) is conducting research in the field of aeronautical human-machine systems. During the development of complex human-machine systems (such as an aircraft), often models of human operators are used in order to optimize system behavior. In contrast to this it is not common in such systems that the machine reasons upon the human's cognitive state during runtime.

In our current main application we regard a helicopter cockpit crew consisting of two persons (cf. figure 1). One of them is the helicopter commander, who is at the same time the operator of a smaller number of Uninhabited Aerial Vehicles (UAVs). The other person is the helicopter pilot. In this context the UBM is developing prototypes of artificial cognitive systems that aid the cockpit crew in coping with high work demands caused by multi-vehicle guidance and mission management (Strenzke et al. 2011). To be more precise, both crew members shall be supported by an assistant system for each workstation. In future, the decisions of such an assistant system shall be based on the current and anticipated situation (e.g. mission progress) as well as on the current and anticipated cognitive state of the operator, which includes his/her beliefs, goals, plan, intended action, interaction with the environment, and subjective workload. The more of this can be assessed, the easier and the earlier human errors can be detected and corrected by assistant system initiative. Multiple approaches to enable an assistant systems to correctly decide about taking initiative (whether, when, and in which way) are currently under research at the UBM. Starting with our Cooperative Automation paradigm for assistant systems, we then give an overview of this research work. Finally, an overarching framework for modeling a human operator is proposed, which is based our Cognitive Process model.

Figure 1: Helicopter Crew in Manned-Unmanned Teaming

Cooperative Automation Approach

The Cooperative Automation approach (Onken and Schulte 2010) is an answer to the vicious circle of automation engineering, which is observable in the evolutionary development of supervisory control systems (Sheridan 1992). In brief, this vicious circle describes the increase of automation to counteract human errors, and thereby in turn provokes human error through automation complexity and opacity. The Cooperative Automation approach is intended to build automation functions that do not accept orders from the human operator in a supervisory control fashion, but instead they work upon the same objectives as the human does in a human-machine-team relationship. According to (Onken and Schulte 2010) a cognitive assistant system shall be designed as such a cooperatively functioning automation. It is furthermore defined by the following basic requirements. The assistant system shall guide the attention of the human operator to the most urgent task. In case the human operator cannot accomplish or should not work upon this task (due to overtaxing, risk or cost), the assistant system shall take initiative to transfer the situation into one which can be handled by the operator (by generating proposals or executing actions on own initiative). Thereby, an assistant system shall provide improvement of the operator's situation awareness, reduction of subjective workload, as well as error avoidance and correction.

Current Assistant System Research

This chapter describes some aspects of the two assistant systems for the helicopter crew.

Knowledge-Based Assistant System

The UAV operator assistant system is realized as a knowledge-based system that supports the UAV operator upon detection or anticipation of suboptimal behavior. It mainly holds knowledge about the modes of interaction with the human operator. See figure 2 and (Donath, Rauschert and Schulte 2010) for further information.

In assistance cases, system has three options to aid the operator. It can provide a warning, suggest an action proposal, or initiate an action (e.g. reconfiguration of some system). To communicate with the operator, the assistant system instantiates a dialog or makes an announcement via speech synthesis and the displaying of a message box in the task-based UAV guidance GUI. Whenever appropriate, this message box includes a few buttons that allow the operator to invoke further aid by the assistance system or to either accept or reject its proposals. See (Strenzke and Schulte 2011) for more detail.

To decide whether, when, and in which way assistance should be provided to the UAV operator

- the assistant system has to be able to anticipate, which tasks the operator has to execute and when he/she is supposed to do this (i.e. plan is incomplete and has to be evolved soon due to time constraints), and
- the assistant system has to be able to notice insufficient quality in the past planning process of the operator (i.e. his/her plan is of too low quality).

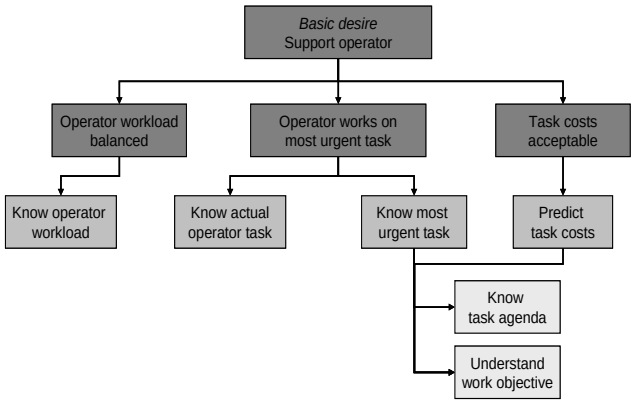


Figure 2: Goal structure of a cognitive assistant system

In principle, the assistant system needs to solve the following questions through target-performance comparison:

- What is the operator planning?
- Is his/her plan good enough?
- Does he/she pursue the plan or make errors?
- Is he/she overtaxed w.r.t. to mental resources?
- What is he/she currently doing?

Clearly, these questions are difficult to answer by a technical system. In the following sections we present our research approaches to these problems.

Mixed-Initiative Mission Planning

The UAV operator generates and modifies multi-UAV mission plans incrementally. The above-mentioned assistant system is able to evaluate, complete, and generate such plans with the aid of the Mixed-initiative Mission Planner (MMP). The MMP is performing plan recognition by planning. In this process it takes into account constraints concerning human goals (which are known by the machine because they are shared in the Cooperative Automation approach), fragments of the human plan, and the current actions of the UAVs. As shown in figure 3, the MMP generates two types of plans: *Reference plans* and *assumed human plans*.

The assumed human plan can be regarded as the assistant system's assumption about the best possible plan the human has in mind after he/she revealed a partial plan[1] by entering UAV tasks into the UAV guidance system (*system*

[1] The human can either be aware of the plan being incomplete or he/she is not, but the machine concludes that the plan is only partial due to missing mission-relevant tasks.

plan, cf. figure 3). The assumed human plan is generated by completing this partial plan by adding the tasks the machine supposed that the human should add to the system plan (at some later point in time). This can be seen as plan recognition by planning. The assumed human plan can be used by the assistant system as a list of tasks to be worked upon by the UAV operator at (or before) the specific times that have been scheduled by the MMP. This can be either the execution of an already planned task (already included in the system plan) or the planning of a task that is missing in the system plan.

The reference plan is generated by the MMP without regarding any human input, simply by solving the UAV mission problem through automated planning. The assumed human plan can be evaluated by comparing its costs to those of the reference plan. In addition to that, the reference plan can be offered by the assistant system to the human in case there is the need of complete mission re-planning. I.e. in case of a difference between the reference and the assumed human plan, the reference plan stands for what the machine supposes the human should do instead of what he/she has planned.

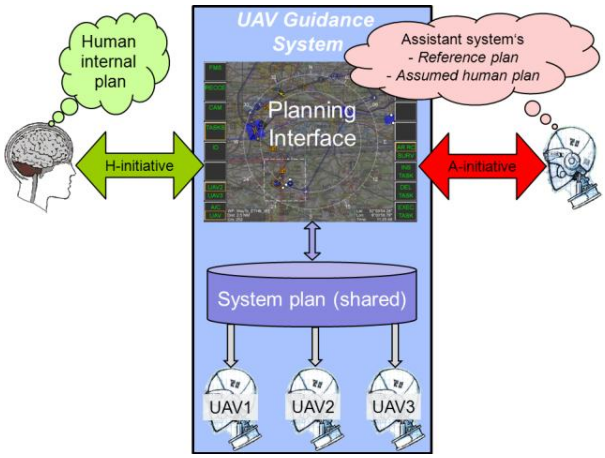


Figure 3: Mixed-initiative planning concept for the MMP

The assistant system has the choice to urge the human operator to make the system plan converge to either the assumed human plan or to the reference plan. The decision of the assistant system can be based either on certain use cases (e.g. major change in the tactical situation or reception of follow-up mission order) or on a cost comparison (target-performance comparison) between the reference and the assumed human plan. Further information about this and on the MMP in general can be found in (Strenzke and Schulte 2011).

The current implementation of the MMP is based on PDDL 2.2 (Edelkamp and Hoffmann 2004) world modeling. We intend to improve the MMP by taking advantage of the expressiveness of PDDL 3.0 (Gerevini and Long 2006) and by plan validation with VAL (Howey, Long, and Fox 2004).

Operator Workload Estimation

In a similar context we experimentally analyzed the correlation between the subjective workload and the behavior of a UAV operator guiding multiple UAVs from a helicopter cockpit, i.e. using them as remote sensor platforms for the reconnaissance of the helicopter route. Especially the changes of his/her behavior can be used as an indicator for high workload situations due to the application of so called self-adaptive strategies (SAS). A human operator will apply suchlike strategies in order to keep the subjective workload within bearable limits and to retard possible performance decrements (Canham 2001; Schulte and Donath 2011). Since such a change in human behavior occurs prior to grave performance decrements, recognizable changes in human operator behavior can be used as trigger for assisting functions. Therefore, an assistant system needs different human operator behavior models, representing behavior within normal workload situations and also models for the behavior modified as a consequence of SAS (cf. figure 4). By the use of such models an assistant system can be enabled to anticipate workload-induced human error and take initiative prior to the occurrence of the error.

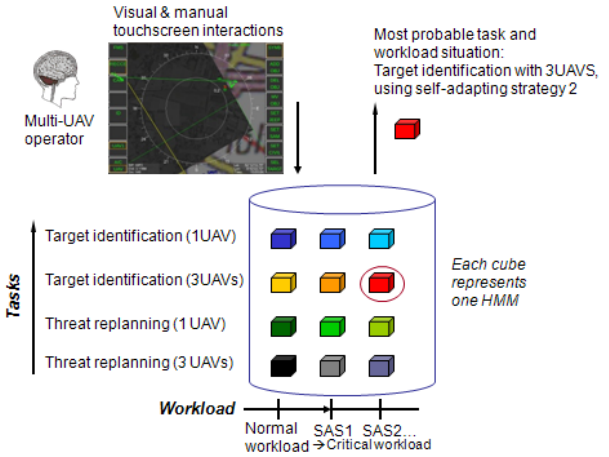


Figure 4: Subjective workload recognition by behavior models

We analyzed gaze tracking and manual interaction (touchscreen button pressing) data to decide if the behavior corresponds with a normal workload situation or a high workload situation. In the experiment human operator behavior was captured during a certain, recurring task situation. The operator had to identify objects (ground vehicles) along the route observed by the reconnaissance UAVs. The object identification task consists of essentially three subtasks, the *recognition and marking of a hotspot* in the photos made by the UAVs, the *object classification process* via the video stream generated by the UAVs, and the *entering the classification result* into the mission management system. Thereby the following SAS were observed (Schulte and Donath 2011):

- Proactive task reduction (e.g. using fewer UAVs for task accomplishment than available)
- Less exect task performance (e.g. watching video stream from a suboptimal UAV/camera position in relation to the object to be identified)
- Omission of subtasks (e.g. classifying the object without entering the result into the system thereafter)
- Complete neglect of the object identification tasks within a complete mission phase
- Purposeful delay of task accomplishment (i.e. interruption / task switching and then continuing)

Each of the mentioned SAS indicates excessive workload and can therefore be used to enable an assistant system to decide to take initiative.

Our approach is to use Hidden Markov Models representing person-specific, task-specific human operator behavior within normal workload conditions in the first step. In future, further HMMs will be added to represent human operator behavior within high workload situations, i.e. the behavioral change of operator behavior caused by the use of SAS (cf. figure 4). See (Donath, Rauschert, and Schulte 2010) for further details. So far, the analysis process is only done offline. Hence, our assistant systems are currently not able to invoke functions based on workload estimation by behavior recognition.

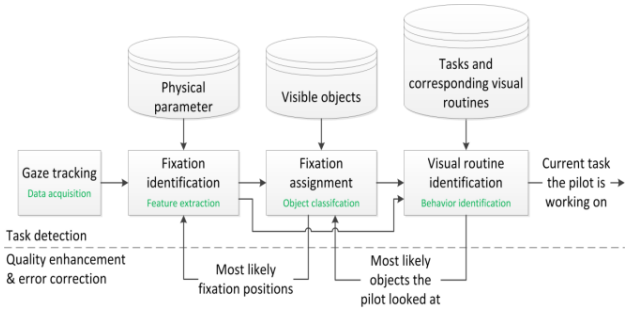
Optimization of Information Channel Selection

In the cockpit setup described above, the pilot flying needs to take over additional responsibilities and tasks from the helicopter commander in order to enable the latter to accomplish the UAV guidance task. Therefore, the pilot flying is also in need of an assistant system, which aids him/her in navigation, aircraft system configuration, and timeliness in the mission plan. This assistant system has to decide whether, when, and through which information channel to inform the pilot about his/her most urgent task and anticipated or detected errors.

As stated before, there is no online workload estimation by behavior patterns in place. Therefore, (Maiwald and Schulte 2011) take the approach to predict the workload on the basis of task models for different task situations and current mental resource demands. By means of a resource model the assistant system is enabled to predict the operator's workload for a certain task situation (see figure 5). Cases of an impending overtaxing can thereby be identified beforehand and prevented by resource-oriented planning of the machine-initiated interactions. E.g. if the pilot is performing a radio transmission his audio processing resources are occupied and therefore the information should be passed on via a message display. But in this case the assistant system also needs to check if the pilot is currently watching the designated display. In addition to that, it would be useful to check if the pilot has noticed or read the corresponding message.

Improving Gaze Tracking Data by Task Context

For the above-mentioned reasons our assistant systems need access to gaze tracking data. In a realistic aircraft cockpit (cf. figure 1) it is very difficult to achieve gaze data quality that is accurate enough to determine in which period of time which object was looked at. This is especially true in the case of online data processing, which is needed for assistant system decision-making. Therefore, the data



quality has to be enhanced by feedback loops as shown in the lower part of figure 6. Its upper part is dedicated to pilot task detection by gaze tracking, which includes the recognition of fixations out of the eye movement data as well as the mapping of fixations to objects in the simulator cockpit (e.g. an aircraft symbol in the moving map display). This data can also be used to make assumptions about the pilot's current situation awareness (e.g. he/she has read a certain message or not) (Maiwald, Benzler, and Schulte 2010).

Figure 6: Using and enhancing gaze data by task context

The data quality improvement and error correction shall be accomplished by applying strategies known from human visual perception, which are the stimulus-based *bottom-up strategy* (Yantis and Jonides 1984) and the knowledge-based *top-down strategy* (Land and Lee 1994).

We intend to use Kalman Filtering for fixation correction towards areas in the display that hold relevant information for the pilot. The detection of the currently processed task by recognizing gaze routines is a candidate for Hidden Markov Modeling. More details about this can be found in (Strenzke et al. 2011).

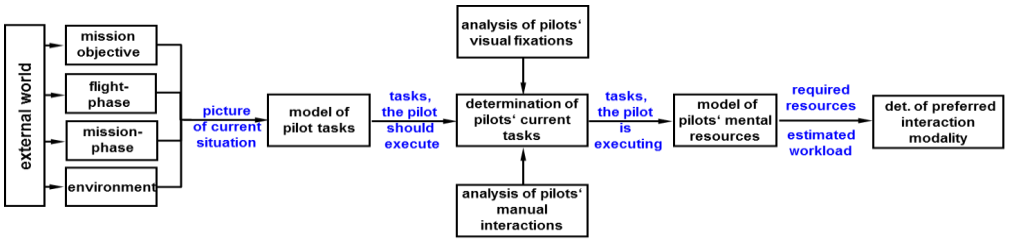


Figure 5: Mental resource-oriented information channel selection for assistant system initiative (Maiwald and Schulte 2011)

A More Complete Human Operator Model

When modeling the behavior of a rational agent, a world model has to be built. An assistant system is such a rational agent, but it has the specialty that it also needs a model of the human operator to be assisted. He/she can be seen as a rational agent as well, because we need the model rather to generate reference behavior (to which the actual measured behavior can be compared), than to predict the actual human behavior. The reason for this is that an assistant system has its focus on recognizing erroneous and suboptimal behavior as well as high workload situations. Therefore it needs to generate a reference behavior dynamically for current situation, to which the actual behavior can then be compared. In contrast to this, the prediction of actual human operator behavior (e.g. human has high workload, will make certain error in near future) is not so important and also much more difficult to cover with a human operator model.

In the previous section it has been shown that there are numerous interdependencies between the operator's goals, plans, actions, behavior, errors and workload. But the current approaches to assess these constructs rely only on simple operator models, which focus only on a specific part of his/her cognitive state and process. A more complete human operator model would allow more thorough reasoning upon the human's cognitive state and cognitive process. E.g. if there are assumptions about his/her goal(s), it is of course easier to assume what he/he is planning. Also, in case there are assumptions about his/her plan, the recognition of the current action is facilitated.

In the following we depict our model of the Cognitive Process (Onken and Schulte 2010) and then add relevant psychological constructs to it in order to enable more sophisticated assistant system decisions upon a model of the human operator's cognitive process.

The Cognitive Process

Figure 7 shows the Cognitive Process that is already used by the UBM to build knowledge-based assistant system behavior as well as cognitive agents (Artificial Cognitive Units, ACUs) for Uninhabited Aerial Vehicle guidance. It has not been used to model the cognitive process of a human operator so far. Due to its dedication to beliefs, goals, plans, and actions (instructions), the Cognitive Process model seems well-suited to unite the above-mentioned approaches to operator modeling and incorporate their interdependencies. A detailed description of the Cognitive Process can be found in (Onken and Schulte 2010).

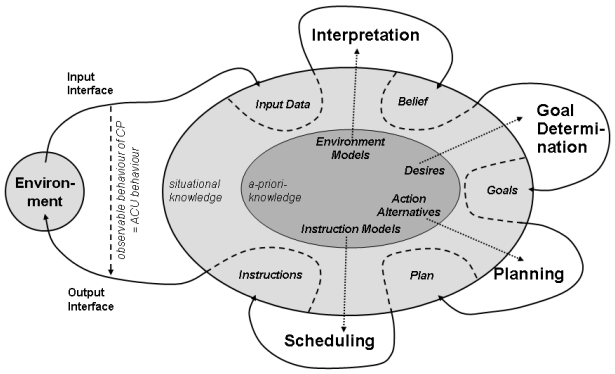


Figure 7: The Cognitive Process (Onken and Schulte 2010)

Adding Behavior, Workload and Error

As we have shown in the overview of our current research topics, a model of a human operator needs to include constructs of *behavior* (which is more than the intended action), mental *workload*, as well as *errors*.

The error taxonomy of (Reason 1990) allows the attribution of error origin and temporality. *Mistakes* are made during planning phase, *lapses* are missed actions (e.g. memory problems), and *slips* refer to errors in the task execution. In Figure 8 we adopt the three psychological constructs into an excerpt of the Cognitive Process model. The mentioned three error types are mapped to the steps in the Cognitive Process as follows. Mistakes happen during planning (e_p), lapses during action selection (e_a), and slips during action execution, i.e. behavior (e_b). Furthermore, errors can take place during information processing (e_i) and goal inference (e_g). Figure 8 shows the enhanced model differentiated by the human operator's conscious internal world and the external world in which his actual behavior (e.g. eye movement, manual interactions) occurs and can be measured. Workload may induce errors (deviations) in the different steps of the cognitive process. Of course, workload is not the only source of human error, but for a start we focus on it in this application-driven model.

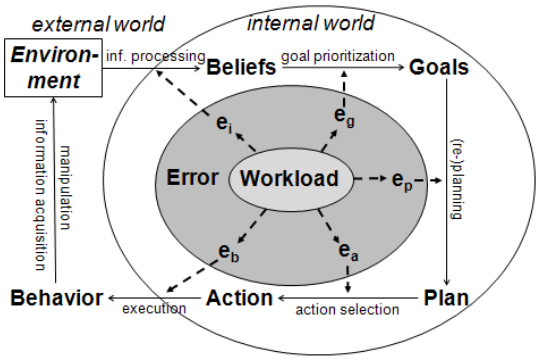


Figure 8: A more complete model of the human operator

Future Work

In future research we will explore implementation approaches for the human operator model presented here. Our novel cognitive system architecture (Cognitive System Architecture with a Central Ontology and Specific Algorithms, COSA²) is based on the Cognitive Process (see figure 7) and able of deliberative planning (Brüggenwirth, Pecher, and Schulte 2011). Integrating a model of the human operator into COSA² could either be achieved by updating the system's beliefs about the cognitive state of the human operator or by adversarial or cooperative planning. The latter means that the machine and the human player have the proposition to solve a problem fully cooperatively and are also able to anticipate each other's behavior. In this case the actions of the human and of the machine can be calculated by means of a uniform planning process. Planning can be seen as top-down determination of plans out of beliefs and goals. The current task or action can be recognized by probabilistic methods as mentioned above (facilitated by assumptions about the human's plan) and then be induced as a constraint into the planning problem. The exploitation of feedback loops as in the mentioned example remains future research work as well.

Acknowledgments

The authors acknowledge the scientific contributions of Diana Donath (UBM), Andreas Benzler and Felix Maiwald ("Military Rotorcraft Associate – Teaming" UBM project staff), as well as Andreas Rauschert ("Manned-Unmanned Teaming" UBM project staff).

References

Brüggenwirth, S.; Pecher, W.; and Schulte, A. 2011. Design Considerations for COSA². In *IEEE Symposium Series on Computational Intelligence (SSCI)*. Paris, France.

Canham, L. S. 2001. Operability Testing of Command, Control & Communications in Computers and Intelligence (C4I) Systems. In *Handbook of Human Factors Testing and Evaluation.* Mallory.

Donath, D.; Rauschert, A.; and Schulte, A. 2010. Cognitive assistant system concept for multi-UAV guidance using human operator behaviour models. In *HUMOUS'10*, Toulouse, France.

Edelkamp, S., and Hoffmann, J. 2004. PDDL2.2: The Language for the Classical Part of the 4th International Planning Competition. In *Technical Report 195, Albert-Ludwigs-Universität Freiburg, Institut für Informatik*, Germany.

Gerevini, A., and Long, D. 2006. Preferences and Soft Constraints in PDDL3. In *Proceedings of ICAPS-06*.

Howey, R., Long, D., and Fox, M. 2004. VAL: Automatic Plan Validation, Continuous Effects and Mixed Initiative Planning using PDDL. *IEEE Conf. on Tools with Artificial Intelligence.*

Land, M. F., and Lee, D.. 1994. Where we look when we steer. In *Nature*, Vol. 369.

Maiwald, F.; Benzler, A.; and Schulte, A. 2010. Berücksichtigung mentaler Operateurzustände bei der Weiterentwicklung wissensbasierter Assistenzsysteme. In *Innovative Interaktionstechnologien für Mensch-Maschine-Schnittstellen*, DGLR Fachausschusssitzung Anthropotechnik. Berlin, Germany.

Maiwald, F., and Schulte, A. 2011. Mental resource demands prediction as a key element for future assistant systems in military helicopters. In *8th Conference on Engineering Psychology & Cognitive Ergonomics*, in conjunction with HCI International.

Onken, R., and Schulte, A. 2010. *System-ergonomic Design of Cognitive Automation: Dual-Mode Cognitive Design of Vehicle Guidance and Control Work.* Heidelberg, Germany: Springer.

Reason, J. 1990. *Human Error*. NewYork, USA: Cambridge University Press.

Schulte, A., and Donath, D. 2011. Measuring Self-adaptive UAV Operators' Load-Shedding Strategies under High Workload. In *8th Conference on Engineering Psychology & Cognitive Ergonomics*, in conjunction with HCI International. Orlando, FL.

Sheridan, T. B. 1992. *Telerobotics, Automation and Human Supervisory Control*. Cambridge, USA: MIT Press.

Strenzke, R., and Schulte, A. 2011. The MMP: A Mixed-Initiative Mission Planning System for the Multi-Aircraft Domain. In *Scheduling and Planning Applications woRKshop (SPARK)* at ICAPS 2011. Freiburg, Germany.

Strenzke, R.; Uhrmann, J.; Benzler, A.; Maiwald, F.; Rauschert, A.; and Schulte, A. 2011. Managing Cockpit Crew Excess Task Load in Military Manned-Unmanned Teaming Missions by Dual-Mode Cognitive Automation Approaches. In *AIAA Guidance, Navigation, and Control (GNC)*. Portland, Oregon.

Yantis, S., and Jonides, J. 1984. Abrupt visual onsets and selective attention: Evidence from visual search. In *Journal of Experimental Psychology: Human Perception and Performance*, Vol. 10.