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

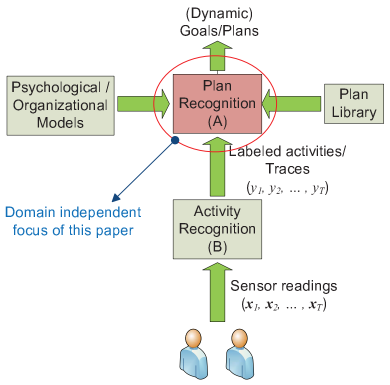
Abstract

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

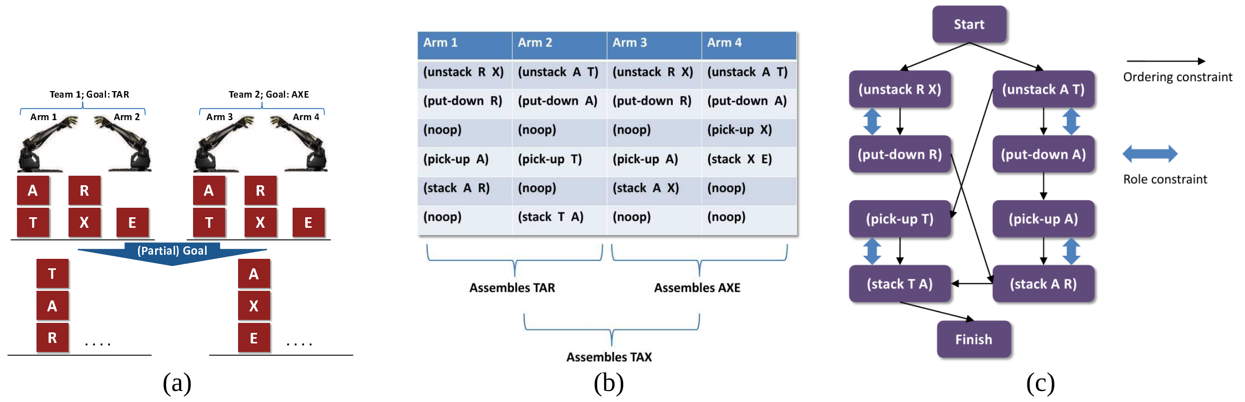
Multi-agent plan recognition (MAPR) refers to the problem of explaining the observed behavior of multiple agents by identifying the (dynamic) team-structures and the team plans (based on a given plan library) being executed, as well as predicting their future behavior. Recently, we introduced a formal 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).


(Dynamic) Goals/Plans
Psychological / Organizational Models
Plan Recognition (A)
Plan Library
Labeled activities/ Traces
$(y_1, y_2, \ldots, y_T)$
Domain independent focus of this paper
Activity Recognition (B)
Sensor readings
$(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_T)$

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,



Arm 1	Arm 2	Arm 3	Arm 4
(unstack R X)	(unstack A T)	(unstack R X)	(unstack A T)
(put-down R)	(put-down A)	(put-down R)	(put-down A)
(noop)	(noop)	(noop)	(pick-up X)
(pick-up A)	(pick-up T)	(pick-up A)	(stack X E)
(stack A R)	(noop)	(stack A X)	(noop)
(noop)	(stack T A)	(noop)	(noop)

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

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

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* $1 \le i \le T,\ 1 \le j \le n$*, there exists* $(p, (k_1, \ldots, k_p, t_p), \pi_p) \in \Pi_{\mathcal{T}|\mathcal{L}}$ *such that* $t_p \le i \le t_p + r(p) - 1$ *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* $f(\Pi_{\mathcal{T}|\mathcal{L}}) \ge k$*?*

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

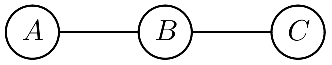
$$f(\{(p_1, o_{p_1}, \pi_1), \ldots, (p_z, o_{p_z}, \pi_z)\}) = \sum_i v(p_i, o_{p_i}, \pi_i),$$

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 • 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 → •END S$, 0, roles$) and (S → •END$, 0, roles$) to $states[0]$
3: for $k \leftarrow 0$ to $L$ do
4:     repeat
5:         ——————Predict——————
6:         for all (Q → … • Y …$, i, r) \in states[k]$ do
7:             add (Y → •$\beta \ldots, k, roles$) to $states[k]$ for all productions in $\mathcal{L}$ with Y on the lefthand side.
8:         end for
9:         ——————Scan——————
10:        for all (Q → … • $\alpha \ldots, i, r) \in states[k]$ 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 → … $\alpha$ • …$, i, z$) to $states[k+1]$
15:            end if
16:        end for
17:        ——————Complete——————
18:        for all (Q → … $\beta$•$, i, r) \in states[k]$ 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 → … $\beta, i+1, k$))
21:                $bestval_k \leftarrow bestval_i + v(x_i^k, o_{xi},$ Q → … $\beta$•)
22:            end if
23:            for all (R → … • Q …$, j, r') \in states[i]$ 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 → … Q • …$, j, z$) 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 • (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). $\square$

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.

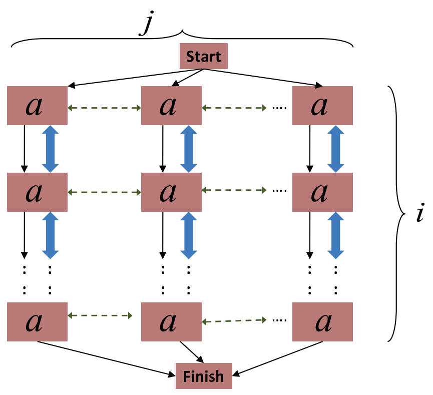


Figure 2: The plan graph corresponding to the $i \times j$ matrix plan. The two-headed dashed arrows represent concurrency constraints, the solid arrows represent ordering constraints, and the two-headed thick arrows represent role constraints.

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-

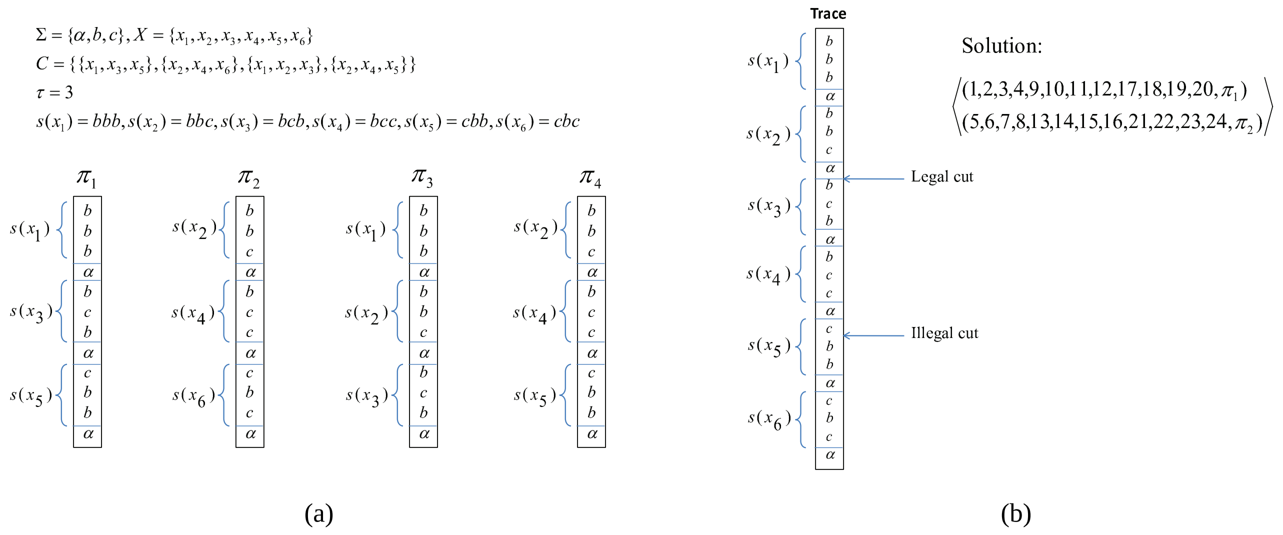

$\Sigma = \{\alpha, b, c\}, X = \{x_1, x_2, x_3, x_4, x_5, x_6\}$
$C = \{\{x_1, x_3, x_5\}, \{x_2, x_4, x_6\}, \{x_1, x_2, x_3\}, \{x_2, x_4, x_5\}\}$
$\tau = 3$
$s(x_1) = bbb, s(x_2) = bbc, s(x_3) = bcb, s(x_4) = bcc, s(x_5) = cbb, s(x_6) = cbc$
π_1
π_2
π_3
π_4
(a)
Trace
Solution:
$\langle (1,2,3,4,9,10,11,12,17,18,19,20,\pi_1)$
$(5,6,7,8,13,14,15,16,21,22,23,24,\pi_2) \rangle$
Legal cut
Illegal cut
(b)

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*.