Multi-Agent Plan Recognition: Formalization and Algorithms

Bikramjit Banerjee and **Landon Kraemer**
School of Computing
University of Southern Mississippi
118 College Dr. # 5106
Hattiesburg, MS 39406
{Bikramjit.Banerjee,Landon.Kraemer}@usm.edu

Jeremy Lyle
Dept of Mathematics
University of Southern Mississippi
118 College Dr. # 5045
Hattiesburg, MS 39406
{Jeremy.Lyle}@usm.edu

Abstract

Multi-Agent Plan Recognition (MAPR) seeks to identify the dynamic team structures and team behaviors from the observations of the activity-sequences of a set of intelligent agents, based on a library of known team-activities (plan library). It has important applications in analyzing data from automated monitoring, surveillance, and intelligence analysis in general. In this paper, we formalize MAPR using a basic model that explicates the cost of abduction in single agent plan recognition by ”flattening” or decompressing the (usually compact, hierarchical) plan library. We show that single-agent plan recognition with a decompressed library can be solved in time polynomial in the input size, while it is known that with a compressed (by partial ordering constraints) library it is NP-complete. This leads to an important insight: that although the compactness of the plan library plays an important role in the hardness of single-agent plan recognition (as recognized in the existing literature), that is not the case with multiple agents. We show, for the first time, that MAPR is NP-complete even when the (multi-agent) plan library is fully decompressed. As with previous solution approaches, we break the problem into two stages: hypothesis generation and hypothesis search. We show that Knuth's “Algorithm X” (with the efficient “dancing links” representation) is particularly suited for our model, and can be adapted to perform a branch and bound search for the second stage, in this model. We show empirically that this new approach leads to significant pruning of the hypothesis space in MAPR.

Introduction

Multi-Agent Plan Recognition (MAPR) seeks an explanation of the observed activity-sequences of a set of intelligent agents, in terms of a given library of team-activities, by identifying the dynamic team structures and team behaviors of the agents. MAPR has important applications in analyzing data from automated monitoring, surveillance, and intelligence analysis in general. In a recent report in the New York Times, it was revealed that in the year 2009, unmanned aerial vehicles collected about 24 years of video data alone from theaters of warfare (Drew Jan 10 2010). Clearly, intelligence analysis needs advances in AI now more than ever, and MAPR constitutes an important sub-problem in this application.

In this paper, we formalize MAPR using a basic model that explicates the cost of abduction in single agent plan recognition by ”flattening” or decompressing the (usually compact, hierarchical) plan library. We show that single-agent plan recognition with a decompressed library can be solved in time polynomial in the input size (particularly, the size of the library), while it is known that with a compressed (by partial ordering constraints) library it is NP-complete (Vilain 1990). This shows that the size of the library plays an important role in the hardness of single-agent plan recognition (as recognized before). Interestingly, however, that is not the case with multiple agents. We show, for the first time, that MAPR is NP-complete even when the (multi-agent) plan library is fully decompressed. Consequently, we seek a branch and bound solution.

As with previous solution approaches, we break the MAPR problem into two stages: hypothesis generation and hypothesis search. We show that Knuth's “Algorithm X” (with the efficient “dancing links” representation) is particularly suited for our model, and can be adapted to perform a branch and bound search for the second stage. We show empirically that this new approach leads to significant pruning of the hypothesis space in MAPR.

Related Work

In this paper, we are more interested in symbolic approaches to plan recognition, and as such do not review the literature on probabilistic approaches. Plan recognition (or more appropriately *keyhole* plan recognition (Cohen, Perrault, and Allen 1981)) has a long and rich history, dating back to the eighties. The problem admits a natural abductive reasoning approach, with some of the earliest work (Kautz and Allen 1986; Charniak and Goldman 1993) emphasizing the central and inherent role of uncertainty manifested by the disconnect between the multiple possible explanations behind the observations, and the true explanation. Kautz and Allen (1986) reduced the problem to deductive inference of explanations for the observations using an action taxonomy, and satisfying a set of simplicity constraints. This formalization showed an approximate correspondence to the vertex cover problem in plan graphs, thus establishing a complexity baseline. Related formalization approaches have encouraged a pre-enumeration of the explanation space before reasoning about observations (Bui 2003). The first significant attempt at establishing the complexity of plan recognition was by Vilain (1990) who transformed Kautz's plan hierarchies to context-free grammars and framed the problem as one of parsing the observation strings (similar approaches were also proposed before (Sidner 1985), while Pynadath and Wellman (2000) proposed probabilistic state dependent grammars to model the recognition of hierarchical behaviors) using the plan grammar. He showed that plan recognition with abstraction and partial ordering in the plan hierarchies is NP-complete, but also identified easier classes under Kautz's scheme. Geib (2004) performed a more recent significant assessment of the complexity of plan recognition under a new model that accommodated issues such as interleaved execution of multiple goals, multiple instantiations of the same goal (such as in a cyber attack), goal abandonment, and partially ordered plan hierarchies (Goldman, Geib, and Miller 1999).

In contrast to plan recognition, multi-agent plan recognition (MAPR) is a much younger area of research. In particular, the hardness of MAPR has not been investigated, which is a major motivation of this work. The significance of considering group actions in order to isolate team plans, rather than a sequential process of recognizing plans of the individual agents separately, has been a repeatedly emphasized theme (Castelfranchi and Falcone 1995; Devaney and Ram 1998; Huber and Durfee 1992; Avrahami-Zilberbrand and Kaminka 2007). In the simpler case where all agents are executing a single plan as one team, the observations can be concatenated and matched against the library (Intille and Bobick 1999). But the more realistic cases involve *dynamic* teams (Tambe 1997), where agents can join and leave teams over the period of the observations. Some previous work has considered static social structures to facilitate the formation of hypotheses on teams and their plans (such as YOYO) (Kaminka and Bowling 2002; Kaminka, Pynadath, and Tambe 2002). Hongeng and Nevatia (2001) eschew group plans and instead use action threads of single agents that are related by temporal constraints generating a multi-agent event graph. Similarly, Avrahami-Zilberbrand and Kaminka (2007) opt for a library of single agent plans instead of team plans, but identify dynamic teams based on the assumption that all agents in a team execute the same plan under the temporal constraints of that plan. While the idea of renouncing a multi-agent plan library is attractive in terms of complexity management, the constraint on the actions of the agents that can form a team can be severely limiting when team-mates can execute coordinated but different behaviors.

More recently, attempts were made to limit the run-time of MAPR by imposing structure (such as temporal dependencies and constraints (Geib 2004; Sukthankar and Sycara 2008)) in the library. Although such structure-based heuristics can limit the hypotheses space, little research has focused on better ways to search this space. For instance, the STABR algorithm (Sukthankar and Sycara 2006) uses sophisticated heuristics to limit the hypothesis space, but uses brute-force to search it. In this paper, we not only establish some hardness results of the basic model of MAPR, but also propose and evaluate a branch and bound algorithm for searching the hypothesis space.

Problem Formulation

Let A be a set of n agents, $\{a_1, a_2, \ldots, a_n\}$. We are given a trace of activities of these agents over T periods of time, in the form of a matrix, $M = [m_{ij}]$, where m_{ij} is the action executed by agent a_j at time i, $j = 1, \ldots, n$ and $i = 1, \ldots, T$. The actions are represented by symbols from an alphabet Σ. We are also given a library (set) of team plans L, where each team plan, $P \in L$, is in the form of an $x \times y$ matrix (where $1 \leq x \leq T, 1 \leq y \leq n$), $P = [p_{ij}]$, where p_{ij} is the action expected from the jth team-member ($j = 1, \ldots, y$) at the ith ($i = 1, \ldots, x$) step from the start of the plan. The actions in the team-plans are also represented by symbols from Σ. A simple example is shown in Figure 1. In this example, the traces of 4 agents' activities over 4 steps are available, as is a library of team plans including at least the 4 plans shown, L_1–L_4. L_1 for example, says that 3 agents in a team execute the (coordinated) sequences cca, aab and bcb.

Definition 1. **(Occurrence)** *A team-plan (sub-matrix) $P = [p_{ij}]_{x \times y}$ is said to* occur *in a matrix M if x contiguous rows ($t_1, \ldots, t_x$, $t_i = t_{i-1} + 1$), and y columns (say $k_1, \ldots, k_y$, a y-selection in any order from n agent indices) can be found in M such that*

$$p_{ij} = M(t_i, k_j), i = 1, \ldots, x, j = 1, \ldots, y$$

We say that each $M(t_i, k_j)$ above is covered *by P.*

For instance, in Figure 1, L_1 occurs in the trace matrix (left), with $(t_1, t_2, t_3) = (2, 3, 4)$ and $(k_1, k_2, k_3) = (4, 1, 2)$. The text at the bottom of this figure describes the occurrences of the other plans. Using these 4 plans (L_1–l_4), the following hypothesis can be created: agents 2 and 1 work in a team during the first time step implementing plan L_3 (in that order), while agents 3 and 4 work individually. At time step 2, agents 4, 2, and 1 form a team for the rest of the observation period to implement plan L_1 (in that order), while agent 3 continues to repeat a pattern of activities (plan L_2) over 2 time steps each, on its own. The utility of this hypothesis is $v(L_1)+2v(L_2)+v(L_3)+v(L_4)$ for some utility function v, and the MAPR problem seeks the maximum-utility hypothesis that explains every observed action *uniquely*.

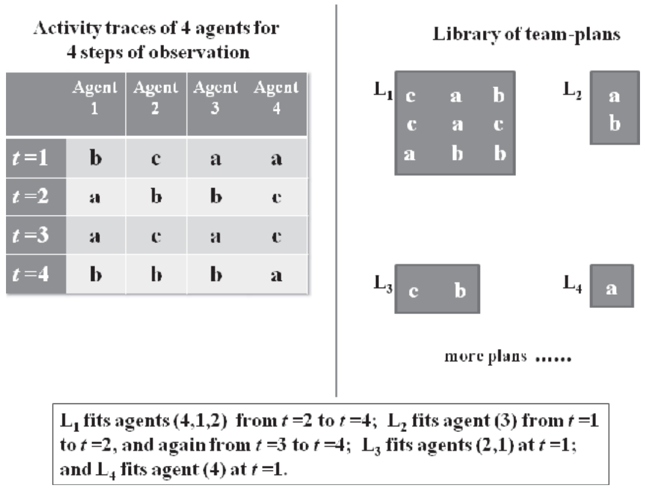
We formalize the above notion of MAPR, parametrized by the sizes of the trace (i.e., T and n) and the library ($|L|$), below:

Definition 2. **(MAPR)** *The multi-agent plan recognition problem, represented as MAPR$(n, T, |L|, k)$ is defined as follows:*

Instance: *Activity matrix M (of size $T \times n$) such that $M(i, j) = m_{ij} \in \Sigma$, a set/library L of sub-matrices, each containing of symbols from Σ, and an integer k.*

Question: *Is there a collection L' of sub-matrices from L, along with their occurrences (Definition 1) such that for each m_{ij} there is exactly one $l \in L'$ that* occurs *in M and* covers *m_{ij}, with $\sum_{l \in L'} v(l) \geq k$?*

Notice that a given $l \in L$ can be used multiple times, e.g., in Figure 1 L_2 is used twice.



	Agent 1	Agent 2	Agent 3	Agent 4
t=1	b	c	a	a
t=2	a	b	b	c
t=3	a	c	a	c
t=4	b	b	b	a

L$_1$ fits agents (4,1,2) from t =2 to t =4; L$_2$ fits agent (3) from t =1 to t =2, and again from t =3 to t =4; L$_3$ fits agents (2,1) at t =1; and L$_4$ fits agent (4) at t =1.

Figure 1: An example of the formalized multi-agent plan recognition problem.

It is worthwhile to note that despite the superficial resemblance between MAPR and TILING (Garey and Johnson 1979) (which is known to be NEXP-complete when the floor-size is expressed compactly), MAPR is computationally easier than TILING. This is because the floor ($n \times n$ matrix) in the TILING problem does not include any information, unlike the $T \times n$ activity trace matrix M in MAPR, and hence its size (n) can be represented compactly in $\log_2 n$ bits, but such sub-linear compaction is generally impossible for the M matrix in MAPR. Therefore, an instance of MAPR is exponentially larger compared to TILING and, as we establish in this paper, is only NP-complete.

Hardness of MAPR

In order to establish the hardness of MAPR, we first present the definitions of a known hard problem, viz., exact cover (EC) (Garey and Johnson 1979).

Definition 3. **(EC)** *The Exact Cover problem is defined as follows:*

Instance: *Set X, and a collection of its subsets, C.*

Question: *Is there a sub-collection $C' \subseteq C$ such that every element of X occurs in exactly one member of C'?*

Lemma 1. *For all $n, T, |L|$, MAPR$(n, T, |L|, k) \in NP$.*

Proof: A proposed solution to MAPR (Definition 2) is specified as a set of tuples $\{(l, k_1, k_2, \ldots, k_y, t)\}$ where $l \in L'$ is a plan of size $x \times y$. Each tuple represents the plan instantiation, complete with its occurrence (Definition 1), i.e., the agent indices $k_1, \ldots, k_y$, and the starting time t in the traces. Verifying a certificate requires checking that

- for each tuple $(l, k_1, k_2, \ldots, k_y, t)$, $l(i+1, j) = M(t+i, k_j)\ \forall i = 0, \ldots x-1, j = 1, \ldots, y$.
- the above mapping covers all elements in M, uniquely. This can be determined simply by maintaining counts.
- $\sum_{l \in L'} v(l) \geq k$

These steps can be performed in time polynomial in $n, T, |L'|$. $\square$

First, we show that plan recognition with a single agent – according to the formulation in Definition 2 – can be solved in polynomial time.

Lemma 2. *MAPR$(1, T, |L|, k)$, i.e., MAPR with just one agent, can be solved in time polynomial in T and $|L|$.*

Proof (sketch): For plan recognition with just a single agent, the plans in the library must pertain to a single agent. Therefore, the plan library consists of a set of sub-strings, and the activity trace of the lone agent is a single string that needs to be partitioned using the sub-string library. In this case, construct a graph by creating a vertex for each instantiation of any plan (sub-string) in the activity trace. Any two instantiations of plans that have a conflict (i.e., possess an overlapping occurrence, according to Definition 1) will be adjacent in this graph. As each instantiation of a plan can be associated with the interval in time for which it occurs, these graphs belong to the well-known class of *interval graphs* (Fulkerson and Gross 1965), with each plan instantiation (sub-string) representing some interval (possibly null) on the activity string. Appropriate weights (details omitted due to lack of space) can be associated to the nodes (plans) based on the plan-values, such that a weighted independent set in this graph will correspond to a valid partition of the trace string. Clearly, this reduction cannot be worse than polynomial in T or $|L|$, and produces an interval graph with input size $O(|L|^2)$. It is known that a maximum weight independent set can be found in an interval graph, in time polynomial in its input size (Hsiao, Tang, and Chang 1992). $\square$

It is important to note that the above result does not necessarily contradict the fact that single agent plan recognition is known to be NP-complete (Vilain 1990). The key to this reconciliation is the fact that previous work in plan recognition assumes that the library of plans is presented in a compact form, usually in hierarchical form (Goldman, Geib, and Miller 1999; Geib 2004; Avrahami-Zilberbrand and Kaminka 2007; Sukthankar and Sycara 2008), such as Hierarchical Task networks (Erol, Hendler, and Nau 1994). In contrast, our formulation of MAPR in Definition 2 uses a flat (matrix) representation of plans. There is a strong justification for using hierarchical models, viz., that we need to identify the higher level semantics behind the ground actions to understand *what* it is that the agent(s) is/are trying to accomplish, and also, possibly, to predict the goal.

Our justification behind using a flat representation for the purpose of formalization comes from the fact that hierarchical representations can always be "flattened", which explicates the worst-case cost of abduction. Hence the flat representation used in Definition 2 is sufficient for formalization, though not necessarily appropriate for solution approaches (except in the worst-case). For instance, Figure 2 (left) shows a hierarchical task network plan in the form of an AND/OR graph (an arc underneath indicates that that node is an AND node meaning all of its successors must be executed, while nodes without arcs underneath are OR nodes), with (partial) ordering constraints (arrows). This model has been extensively used in the past (Avrahami-Zilberbrand and

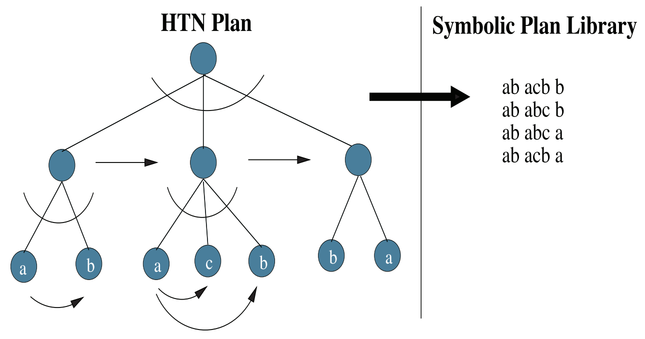


Figure 2: An illustration of the transformation of an HTN plan into 4 symbolic plan strings in the "flat" representation.

Kaminka 2007; Sukthankar and Sycara 2008). The right part of Figure 2 shows the corresponding symbol strings that result from flattening it, and constitute the plan library for a single agent. It is clear from Figure 2 that flattening results in an exponential increase in the size of the plan library, more so with fewer ordering constraints. Thus $|L|$ in our formulation is potentially intractably sized, compared to the (compact) hierarchical representation.

The above discussion explicates how important a role $|L|$ plays in determining the hardness of single agent plan recognition problems, and this has been recognized in the existing literature (Vilain 1990; Geib 2004). However, as we now show in our main result below, this is not the case with multiple agents. General MAPR is hard irrespective of how compactly L is represented.

Theorem 3. *Even with the flat representation of plans, MAPR$(n, T, |L|, k)$ as given in Definition 2 is NP-complete.*

Proof (sketch): To prove that MAPR is NP-hard, a general instance of EC (with S and C) from Definition 3 can be polynomially reduced to a special case of MAPR where $n = |S|$ and T is such that $T < n$ but $|\Sigma|^T \geq n$. Then for each element of S we create a distinct string of length T by sampling with replacement from Σ. This produces n strings of length T each, and these together produce the trace matrix M. These strings can then be used to create a matrix of size $T \times |c|$ for each $c \in C$, thus producing the plan library, L of size $|L| = |C|$. It can be shown that a solution to this instance of MAPR exists if and only if a solution exists to EC. Together with Lemma 1, this completes the proof. □

The results below follow from the properties of the EC problem:

Corollary 4. *MAPR as given in Definition 2 remains NP-complete*

- *even if all plans in L describe the behaviors of only 3-agent teams;*
- *even if no string of activities of any agent is present in more than 3 plans in L.*

Algorithms

We now turn to solving MAPR in the decompressed model. Since we have shown reducibility of EC to MAPR, we exploit a known approach to EC, viz., Knuth's Algorithm X with the efficient "dancing links" (DLX) (Knuth 2000) representation. It solves EC for any instance EC(X, C) represented as a Boolean matrix $E = [e_{ij}]$, where e_{ij} is 1 iff $c_i \in C$ contains X_j for $i = 1, \ldots, |C|$ and $j = 1, \ldots, |X|$. Algorithm X finds a set of rows of E such that for every column of E, there is exactly one row that contains a 1, by backtracking search using the efficient dancing links representation (see (Knuth 2000); details omitted due to lack of space).

We can convert any instance of MAPR with a trace M and plan library L to a Boolean matrix E as follows. E shall have Tn columns, and a row for each occurrence (Definition 1) of each plan in L. We use the notation $r(l)$ to denote the number of rows in a matrix l and $c(l)$ the number of columns in it. For a given occurrence of $l \in L$, $(l, k_1, k_2, \ldots, k_{c(l)}, t)$, we can create a row in E as

$$e_{row,col} = \begin{cases} 1 & \text{if } col = (t+i-1)n + k_j, \text{ for some} \\ & i \in [0, r(l)-1], j \in [1, c(l)] \\ 0 & \text{otherwise} \end{cases}$$

Algorithm 1 describes this process of creating E in greater detail, and the small example shown in Figure 3 illustrates it, where each plan L_i in the library yields exactly one occurrence corresponding to the ith row in E.

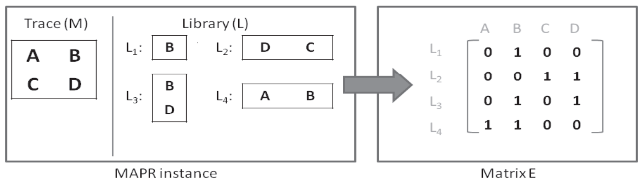


Figure 3: An illustration of the transformation of a MAPR instance into the matrix E, by Algorithm 1.

One important benefit of using Algorithm X to solve MAPR is that it provides us with search tree which, in turn, allows us to prune sub-trees from our search with *branch and bound*. Algorithm 2 shows our modified version of Knuth's Algorithm X, adapted for branch and bound, to find an exact cover of the matrix E constructed by Algorithm 1. In order to perform branch and bound we need three important functions. First, let $h(i)$ be a function which maps a row in the matrix E to the plan from which it was generated. Then, if S is a whole or partial solution (i.e. a list of rows in E), the value of S is given by

$$\texttt{Value}(S) = \Sigma_{s \in S}\, v(h(s))$$

Most importantly, for branch and bound we need a function that gives us an upper bound on the value of the remaining part of any full solution (of which S is a part) that we can achieve by traversing any given sub-tree. In Algorithm 2, we refer to this function as `BestPossible`. `BestPossible` can have a variety of definitions producing different tightnesses of the upper bound; however, in our experiments, we specifically used

$$\texttt{BestPossible}(E) = \sum_{j=1}^{c(E)} \max_{i \in r(E)} \{v(h(i)) | E(i,j) = 1\}$$

Algorithm 1 Build-E(M, L)

```
Input: A trace matrix $M$ (of size $T \times n$) and a plan library $L$.
Output: A Boolean matrix $E$.
  $E \leftarrow \emptyset$
  for each plan $P \in L$ do
    $x \leftarrow$ number of rows of $P$, i.e., $r(P)$
    $y \leftarrow$ number of columns of $P$, i.e., $c(P)$
    for $r \leftarrow 1 \ldots T - x + 1$ do
      for $c \leftarrow 1 \ldots n$ do
        $\mu_c \leftarrow \begin{bmatrix} m_{r,c} \\ m_{r+1,c} \\ \vdots \\ m_{r+x-1,c} \end{bmatrix}$
      end for
      for each unordered selection of $y$ indices $(i_1, i_2, \ldots, i_y)$ from the agent set $\{1, 2, \ldots, n\}$ such that $\begin{bmatrix} \mu_{i_1} & \mu_{i_2} & \ldots & \mu_{i_y} \end{bmatrix} = P$ (based on $\mu_c$'s computed in the previous step) do
        $A \leftarrow 0_{1 \times (Tn)}$
        for $i' \leftarrow i_1 \ldots i_y$ do
          for $j' \leftarrow 0 \ldots x - 1$ do
            $A(j'n + i') \leftarrow 1$
            $E \leftarrow \begin{bmatrix} E \\ A \end{bmatrix}$
          end for
        end for
      end for
    end for
  end for
  Return $E$
```

because DLX allows us to find $\max_i\{v(h(i))|E(i,j) = 1\}$ in constant time. This is because the rows in E are already sorted by v, so that $v(h(1)) \geq v(h(2)) \geq \ldots \geq v(h(r(E)))$, in the hope of finding a high-valued solution early to be able to bound more solutions later. It can be shown (proof omitted) that Value(S)+BestPossible(E) always exceeds the highest utility solution that could possibly match S, and is therefore a feasible pruning criterion.

The function ChooseColumn returns the next column that should be covered. It can be defined in variety of ways, but (Knuth 2000) reports that choosing the column with the fewest number of ones reduces the branching factor quite effectively. In our experiments we have used this heuristic.

Evaluation

In order to evaluate the effectiveness of our pruning method, we have run our algorithms on a series of random instances of MAPR problems. In generating each random instance, we first generate a random trace with dimensions $T \times n$, with each element of the trace belonging to the set of possible actions, Σ. Then, we randomly partition the trace matrix, adding the sub-matrices to the plan library L, so that at least one possible solution exists, and then add z random new plan templates to the library. In our experiments, we have generated 2400 such instances with $T = 100$, $n = 20$, $|\Sigma| = 10$, and $z = 50$.

Algorithm 2 Search(E, S)

```
  if $E = \emptyset$ then
    if $best\_solution\_value <$ Value($S$) then
      $best\_solution\_value \leftarrow$ Value($S$)
      $best\_solution \leftarrow S$
    end if
    Return
  end if
  $best\_possible \leftarrow$ Value($S$) + BestPossible($E$)
  if $best\_solution\_value > best\_possible$ then
    Return
  end if
  $C \leftarrow$ ChooseColumn($E$)
  for each $row$ such that $E(row, C) = 1$ do
    $S \leftarrow S \cup \{row\}$
    $B \leftarrow E$
    for each $col$ such that $E(row, col) = 1$ do
      Remove $col$ from $B$.
      for each $r$ such that $E(r, col) = 1$ do
        Remove $r$ from $B$
      end for
    end for
    Search($B, S$)
    $S \leftarrow S \setminus row$
  end for
```

As a metric for the amount of work done in finding solutions, we use Knuth's concept of *updates* (Knuth 2000), where an update is the removal and reinstatement of an object in his DLX implementation.

Let the sequence of solutions found by the unpruned search be represented as $S = \{s_1, s_2, \ldots, s_j\}$, then the sequence of solutions found by the pruned search, S_P, must be a sub-sequence of S. Then let $W_{P_{s_i}}$ and W_{s_i} represent the work required to find the solution s_i with and without pruning, respectively. Since, pruned and unpruned search traverse the search tree in the same manner, then $W_{P_{s_i}}$ and W_{s_i} are directly comparable for $s_i \in S_P$.

In our experiments, for each randomly generated MAPR instance we take the sequence S_P and partition it into sub-sequences $S_{P_0}, S_{P_1}, \ldots, S_{P_{\lceil log_{10} s_j \rceil}}$ such that $s_i \in S_{P_l} \Rightarrow \lfloor log_{10} i + .5 \rfloor = l$. Note that S_{P_l} might be empty. Now let $s_l^* = s_i \in S_{P_l} | s_j \in S_{P_l} \Rightarrow j \leq i$. Figure 4 shows the average of $\left(\frac{W_{P_{s_l^*}}}{W_{s_l^*}} \right)$ over 2400 runs, with 95% confidence intervals, for $0 \leq l \leq 9$.

Figure 4 suggests that as the size of the solution space increases, the amount of pruning tends to increase. This is due to the fact that instances with more solutions tend to be less constrained than those with fewer solutions, resulting in deeper trees with more branching. This increases the likelihood that pruning a sub-tree will significantly reduce the amount of work done. Furthermore, as more solutions are found the value of the current best solution cannot decrease, which, in turn, increases the chances that sub-trees can be

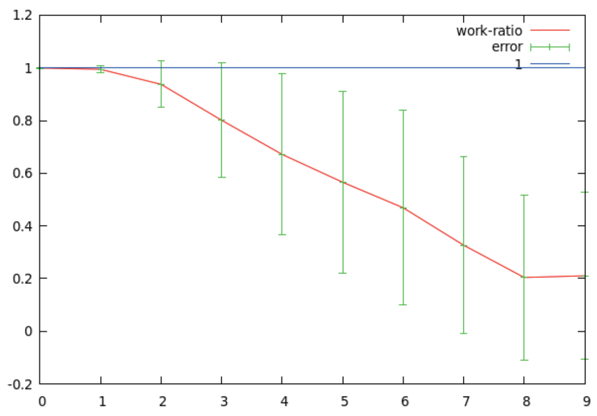


Figure 4: Plot of averages of the work-ratio between pruned and unpruned searches, against the number of solutions seen by the unpruned search on the log-scale. Ratio 1 is shown for reference.

pruned.

While this experiment demonstrates that significant pruning is possible with our algorithm, it is likely that better or worse pruning may result in instances with a different set of parameters. We leave it as future work to study the relationship between the parametric settings $(n, T, |\Sigma|, |L|)$ of MAPR instances and the effectiveness of the pruning algorithm, and also to develop improved pruning techniques.

Conclusion

We have formalized MAPR using a new model and presented its hardness, revealing an important fundamental distinction between the hardnesses of single and multi-agent plan recognition. We have also proposed a first-cut approach to solving MAPR in this model, and have evaluated its efficacy. This model of MAPR is clearly capable of handling dynamic team memberships and behavioral patterns, although complex features – such as goal abandonment and interleaved plan execution – will require modifications to Algorithm 1 that will make it significantly more complex. Managing this complexity in the extended model is an important future direction.

Acknowledgements: We appreciate the feedback from the anonymous reviewers. 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 AAAI conference*.

Bui, H. 2003. A general model for online probabilistic plan recognition. In *Proc. of the International Joint Conference on Artificial Intelligence (IJCAI*, 1309–1315.

Castelfranchi, C., and Falcone, R. 1995. From single-agent to multi-agent: Challenges for plan recognition systems. In *Proceedings of the IJCAI-95 Workshop on The Next Generation of Plan Recognition Systems*, 24–32.

Charniak, E., and Goldman, R. 1993. A bayesian model of plan recognition. *Artificial Intelligence* 64:53–79.

Cohen, P.; Perrault, C.; and Allen, J. 1981. *Strategies for Natural Language Processing*. Lawrence Earlbaum Assoc. chapter Beyond question answering.

Devaney, M., and Ram, A. 1998. Needles in a haystack: Plan recognition in large spatial domains involving multiple agents. In *Proceedings of AAAI conference*.

Drew, C. (Jan-10) 2010. Military is deluged in intelligence from drones. In *New York Times*.

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.

Fulkerson, D. R., and Gross, O. A. 1965. Incidence matrices and interval graphs. *Pacific Journal of Mathematics* 15:835–855.

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.

Geib, C. 2004. Assessing the complexity of plan recognition. In *Proc. of AAAI-04*.

Goldman, R. P.; Geib, C. W.; and Miller, C. A. 1999. A new model of plan recognition. *Artificial Intelligence* 64:53–79.

Hongeng, S., and Nevatia, R. 2001. Multi-agent event recognition. In *Proceedings of the Eighth IEEE International Conference on Computer Vision*, 84–91 vol.2.

Hsiao, J. Y.; Tang, C. Y.; and Chang, R. S. 1992. An efficient algorithm for finding a maximum weight 2-independent set on interval graphs. *Information Processing Letters* 43(5):229–235.

Huber, M., and Durfee, E. 1992. Plan recognition for real-world autonomous robots: Work in progress. In *Working notes of AAAI symposium: Applications of AI to Real-World Autonomous Robots*.

Intille, S., and Bobick, A. 1999. A framework for recognizing multi-agent action from visual evidence. In *Proc. of AAAI*.

Kaminka, G., and Bowling, M. 2002. Towards robust teams with many agents. In *Proc. AAMAS-02*.

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*.

Knuth, D. E. 2000. Dancing links. In Davies, J.; Roscoe, B.; and Woodcock, J., eds., *Millennial Perspectives in Computer Science: Proceedings of the 1999 Oxford-Microsoft Symposium in Honour of Sir Tony Hoare*, 187–214.

Pynadath, D. V., and Wellman, M. P. 2000. Probabilistic state-dependent grammars for plan recognition. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence, UAI2000*, 507–514. Morgan Kaufmann Publishers.

Sidner, C. 1985. Plan parsing for intended response recognition in discourse. *Computational Intelligence* 1(1):1–10.

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. 1997. Towards flexible teamwork. In *Journal of Artificial Intelligence Research*, volume 7, 83–124.

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