

Branch and Price for Multi-Agent Plan Recognition

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

Abstract

The problem of identifying the (dynamic) team structures and team behaviors from the observed activities of multiple agents is called Multi-Agent Plan Recognition (MAPR). We extend a recent formalization of this problem to accommodate a compact, partially ordered, multi-agent plan language, as well as complex plan execution models – particularly plan abandonment and activity interleaving. We adopt a branch and price approach to solve MAPR in such a challenging setting, and fully instantiate the (generic) pricing problem for MAPR. We show experimentally that this approach outperforms a recently proposed hypothesis pruning algorithm in two domains: multi-agent blocks world, and intrusion detection. The key benefit of the branch and price approach is its ability to *grow* the necessary component (occurrence) space from which the hypotheses are constructed, rather than begin with a fully enumerated component space that has an intractable size, and search it with pruning. Our formulation of MAPR has the broad objective of bringing mature Operations Research methodologies to bear upon MAPR, envisaged to have a similar impact as mature SAT-solvers had on planning.

Introduction

Multi-agent plan recognition (MAPR) refers to the problem of explaining the observed behavior trace 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. Since multiple explanations (and corresponding predictions) are possible, a traditional approach has been to prune in the space of explanations/hypotheses to arrive at the most likely hypothesis (or possibly multiple highly likely ones) (Sukthankar and Sycara 2008; Banerjee, Kraemer, and Lyle 2010). We seek a better alternative to this approach, but we require a sufficiently general model of MAPR to accomplish this in a principled way.

Recently, a formal model for MAPR was proposed and used to investigate the complexity of its simplest setting (Banerjee, Kraemer, and Lyle 2010). However, this model accommodates neither an expressive plan language, nor complex plan execution models such as *plan abandonment* (agents starting a plan before completing an earlier

plan) and *activity interleaving* (agents interrupting a plan to serve a different plan, and resuming the earlier plan again, possibly repeatedly). We extend this model to accommodate these features, and adopt an OR technique –*branch and price* (Wolsey 1998)–to solve MAPR in this extended model. The key result of this paper is that a hypothesis *growth* approach (afforded by branch and price) is a more scalable alternative to the traditional hypotheses pruning approach.

We begin with an illustration of MAPR in a multi-agent blocks world domain, shown in Figure 1. In part (a), two teams of robotic arms assemble (i.e., spell out) 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 available to the (remote) recognizer, who is not aware of the team-structure (i.e., the mapping of agent-id to stack-id). This assumption partly models the realistic incomplete information under which the recognizer must operate. While arms 1 and 2 appear to jointly assemble “TAR”, and arms 3 and 4 appear to jointly assemble “AXE”, arms 2 and 3 seem to assemble “TAX” as well, creating ambiguity for the recognizer. The key insight is to *partition* the trace into non-overlapping team plans, such that invalid teams (such as the supposed team of agents 2 and 3) fail to yield a complete partition hypothesis. In this example, agents 1 and 4 would be executing illegal plans individually, or building separate stacks as a team, neither of which yields a valid partition hypothesis. Note, teammates are not required to start plan execution at the same time (unlike what Figure 1 (b) might suggest), 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 for multi-agency: *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 time and the team size needed to execute a plan are unspecified but constrained, e.g., between 1 and 4 agents can (collaboratively) execute this plan, in 5 to unlimited time steps (due to noops).

We do not claim the above simple multi-agent plan lan-

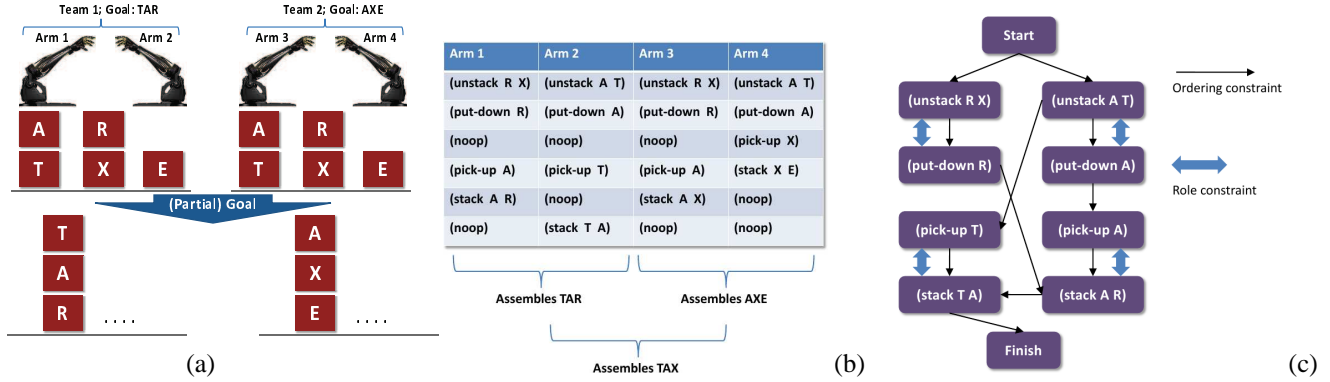


Figure 1: Multi-Agent Blocks World. Agents and teams with partially specified goals (a), trace (b), and a plan (c).

guage to be ideal, and in fact, an ideal language would be a separate topic altogether. We use this language because it is conformant to the existing planning literature and has minimal additions to accommodate multi-agency. It is not as compact as (Sukthankar and Sycara 2008), but it offers more flexible and compact definitions of multi-agent plans than (Banerjee, Kraemer, and Lyle 2010). We use a library of such plans to partition the trace thus identifying individual (dynamic) teams and their plans, whether they are complete or incomplete, interleaved or non-interleaved. We formulate MAPR in this extended model as a branch and price problem and show that this leads to a more scalable solution than hypotheses pruning, in two experimental domains.

We address an *idealized* setting of MAPR that ignores noisy (one observation confused for another) and missing observations, and incompleteness of the library, besides abstracting away the process of *activity recognition* (Sadilek and Kautz 2010) whereby the observed abstract activities in the trace are determined from raw sensor data. It is our hope that identifying the most appropriate solution approach for the idealized problem will lead to useful insights to ultimately address non-idealized settings in a systematic manner. In particular, we expect that the present insight of *growing* hypotheses being a more efficient alternative to *hypotheses pruning* in MAPR, will spur a new set of heuristic approaches for non-idealized settings, to verify whether the benefit extends there.

Preliminaries

Let A be a set of n agents, $A = \{1, 2, \dots, n\}$, 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, T , of observed activities of these agents over T steps of time, in the form of a $T \times n$ matrix, $T = [t_{ij}]$, where $t_{ij} \in \Sigma$ is the action executed by agent j at time i , $j = 1, \dots, n$ and $i = 1, \dots, T$. We are also given a library of team plans \mathcal{L} , which 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 Net-

work (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. This perspective resembles the generative model of plan execution underlying PHATT (Geib, Maraist, and Goldman 2008), albeit for multiple agents.

Definition 1 (Plan) A multi-agent plan, π is given by a tuple $\pi = \langle S, O, R, C \rangle$, where

- $S = \{s_1, s_2, \dots\}$ is a set of plan steps (i.e., vertices of graph, as in Figure 1(c)),
- $O = \{(s_i, s_j), \dots\}$ is a set of ordering constraints, stating that s_i must be executed before s_j ,
- $R = \{(s_i, s_j), \dots\}$ is a set of role constraints, stating that s_i and s_j must be executed by the same agent,
- $C = \{(s_i, s_j), \dots\}$ is a set of concurrency constraints, stating that s_i and s_j must be executed concurrently.

Additional constraints can be incorporated if needed. Note that the role and concurrency constraints can be of positive and negative kinds, the latter identifying pairs of steps that must *not* be performed by the same agent, or concurrently. But the experimental domains in this paper will only need the positive kind of constraints, so we do not consider the negative kind any further.

Definition 2 (Occurrence) An occurrence of a plan $\pi = \langle S, O, R, C \rangle$ in trace T is given by a set of triples $o_\pi = \{(s, k, t), \dots\}$ such that

- $(s, k, t) \in o_\pi \Rightarrow 1 \leq t \leq T \wedge k \in A \wedge s = T(t, k) \in S$
- $\forall (s_i, s_j) \in O, (s_i, k_i, t_i) \in o_\pi \wedge (s_j, k_j, t_j) \in o_\pi \Rightarrow t_i < t_j$
- $\forall (s_i, s_j) \in R, (s_i, k_i, t_i) \in o_\pi \wedge (s_j, k_j, t_j) \in o_\pi \Rightarrow k_i = k_j$
- $\forall (s_i, s_j) \in C, (s_i, k_i, t_i) \in o_\pi \wedge (s_j, k_j, t_j) \in o_\pi \Rightarrow t_i = t_j$

Additionally, if $S = \{s_i | (s_i, k_i, t_i) \in o_\pi\}$, then o_π is said to be a complete occurrence. Otherwise, it is an incomplete occurrence, and could either have been abandoned or might be completed after T .

For instance, the only complete occurrence of the plan in Figure 1(c) in the trace in 1(b) is $o_{TAR} = \{((\text{unstack R X}), 1, 1), ((\text{unstack A T}), 2, 1), ((\text{put-down R}), 1, 2), ((\text{put-down A}), 2, 2), ((\text{pick-up A}), 1, 4), ((\text{pick-up T}), 2, 4), ((\text{stack A R}), 2, 4)\}$.

R), 1, 5), ((stack T A), 2, 6)}. Similarly, complete occurrences of plans with goals “AXE” and “TAX” can be created, as indicated in Figure 1(b). As an illustration of an incomplete occurrence for goal “AXE”, o'_{AXE} , consider the team of agents $\{1, 4\}$ and the first 4 steps of observation in Figure 1(b). This is given by $o'_{AXE} = \{((\text{unstack R X}), 1, 1), ((\text{unstack A T}), 4, 1), ((\text{put-down R}), 1, 2), ((\text{put-down A}), 4, 2), ((\text{pick-up A}), 1, 4), ((\text{pick-up X}), 4, 3), ((\text{stack X E}), 4, 4)\}$.

The set of agents involved in an occurrence o is a (hypothesized) *team*, and represented as $X^o = \{k | (s, k, t) \in o\}$. Also, let $t_{\max}^o = \max\{t | (s, k, t) \in o\}$ be the finish time of an occurrence. For the case that the observed agents are *not* interleaving plan execution, for each occurrence o_π that is incomplete, if $t_{\max}^{o_\pi} = T$, or all agents in X^{o_π} only execute “noop”s between $t_{\max}^{o_\pi}$ and T , then o_π is assumed a potential candidate for completion after T and yields a *prediction* (stating that the set of agents in X^{o_π} must accomplish the unexecuted steps of π after time T). Otherwise, o_π is assumed to be abandoned. On the other hand, when the observed agents are indeed interleaving plan execution, then no such distinction can be made for incomplete occurrences. Moreover in this case, prediction of observations beyond T is more complex, with multiple reasonable alternatives. In this paper, we only focus on the problem of *explanation*, and defer a general treatment of prediction in both interleaving and non-interleaving cases to future work.

Definition 3 (Interleaved Explanation) *An interleaved explanation of \mathcal{T} is a set of occurrences $\Pi = \{o^1, o^2, \dots\}$ such that all of the following hold:*

Library-conformity: *Each $o^i \in \Pi$ is a (complete or incomplete) occurrence of some plan $\pi \in \mathcal{L}$,*

Non-overlap: $\forall o^i, o^j \in \Pi, o^i \cap o^j = \emptyset,$

Coverage: $\forall (i, j) \text{ s.t. } \mathcal{T}(i, j) \neq (\text{noop}), \exists o^k \in \Pi \text{ s.t. } (\mathcal{T}(i, j), j, i) \in o^k.$

The above definition requires that Π *partitions* \mathcal{T} . For instance, if a complete list of (complete and incomplete) occurrences were generated for the example in Figure 1 (assuming no other start state is possible, and the only goals possible are “TAR”, “TAX” and “AXE”), the only subset of those occurrences that successfully partitions \mathcal{T} would be $\Pi = \{o_{TAR}, o_{AXE}\}$. Thus the remaining occurrences o_{TAX} , o'_{AXE} and others are effectively discarded as being contradictory in explaining \mathcal{T} . A valid partition (not necessarily unique) yields a self-consistent explanation. Notice that in this example, the occurrences actually fit into the partition/explanation in a non-interleaved manner, although they also satisfy the above definition. In order to capture non-interleaved partition/explanation, we require an additional constraint as stated in the following definition.

Definition 4 (Non-interleaved Explanation) *A non-interleaved explanation of \mathcal{T} is a set of occurrences $\Pi = \{o^1, o^2, \dots\}$ such that in addition to the conditions in Definition 3, the following holds for each $o^j \in \Pi$*

$$\forall t \in [t_{\min}^{o^j}, t_{\max}^{o^j}], \forall i \in X^{o^j}, (\mathcal{T}(t, i), i, t) \in o^j \vee \mathcal{T}(t, i) = (\text{noop})$$

where X^{o^j} and $t_{\max}^{o^j}$ are as defined before, and $t_{\min}^{o^j}$ is similarly defined as $t_{\min}^{o^j} = \min\{t_i | (s_i, k_i, t_i) \in o^j\}$.

For an abandoned team plan, this definition requires all teammates to abandon it at the same time. We do not address non-unanimous abandonment in this paper. We call the set of possible explanations of \mathcal{T} , \mathcal{P} . Alternative explanations in \mathcal{P} are also called *hypotheses*, and the goal of MAPR is either to output the *most likely* hypothesis, or a set of hypotheses that are most highly ranked in some way. Our search approach accommodates both choices, albeit in limited ways. We associate a utility function $f : \mathcal{P} \mapsto \mathbb{R}$ to the explanations/hypotheses, so that each explanation of \mathcal{T} can be evaluated for its preferability, with higher f indicating greater preferability. f can be defined in various ways; e.g., setting $f(\{o^1, \dots, o^z\}) = -z$ captures Kautz & Allen’s “minimal top-level plans” (1986) criterion for a single agent, and Kaminka & Bowling’s “maximum coherence” (2002) for multiple agents. In this paper, we define f to represent the log-likelihood of an explanation using a simple probability model, and assuming independence of occurrences. Although this returns one hypothesis, our search procedure can be easily modified to return all hypotheses that exceed a selected f value, thus accommodating the other goal of MAPR. The following is a straightforward extension of the hardness result of (Banerjee, Kraemer, and Lyle 2010), and stated without proof:

Theorem 1 *For an arbitrary trace \mathcal{T} and a library of plan graphs \mathcal{L} , the problem of deciding whether a non-interleaved explanation of \mathcal{T} exists is NP-complete.*

Our experimental results indicate that finding an interleaved explanation is even harder.

Solution Approach

The approach suggested in (Banerjee, Kraemer, and Lyle 2010) segregates the occurrence generation from hypothesis search, and as such, must completely enumerate the occurrence space. However, this space can be large, with $O(T^{22^n})$ being a tight bound on its size. Moreover, only a small part of this space will ever appear in the solution, since the number of activities to be explained is only $O(Tn)$. Enumerating the occurrences before the search can begin is a significant liability, not because it is expensive (it is not, compared to the search time), but because pruning oriented search is simply inefficient in the hypothesis space that is grounded on a complete occurrence space (components of hypotheses), as our experiments show. Other existing approaches have also emphasized on pruning hypotheses (explanations that are built out of the set of occurrences), albeit heuristically (Sukthankar and Sycara 2008). This paper is partly motivated by the need for an alternative perspective to hypotheses search.

In this paper, instead of pruning unwanted hypotheses from a (potentially) very large space, we *grow* incrementally the set of occurrences most likely to be used in the most valued hypothesis, and only ever generate those occurrences. We adopt a branch and price (Wolsey 1998) approach for hypothesis search, with *column generation* used for incre-

mental occurrence generation and incorporation in the solution basis. As customary in branch and price, we establish a master problem (MP) which is the original MAPR problem posed as an integer (boolean) program, but relaxed to be solved efficiently as a linear program. Initially, its basis only consists of occurrences (columns) that are sufficient to produce a feasible solution to the LP relaxation. This is called a restricted master problem (RMP). Then, column generation is used as secondary (or slave problem, SP) to the master to identify a new occurrence (column) to enter the basis, and this process repeats until the solution to the RMP is guaranteed to also be a solution to the MP. Although such a solution may not solve the original integer program (because it may assign non-integers to some variables), it provides an upper bound that is useful for branch and bound search. More importantly, the alternation of RMP and SP at each node spread over paths in the branch and bound tree allows us to *incrementally* generate new hypotheses, i.e., to grow hypotheses.

Branch and Price Formulation

As in (Banerjee, Kraemer, and Lyle 2010), let E be a binary matrix which has Tn columns and $O(T^2 2^n)$ rows. In the example of Figure 1, assuming only the three possible goals shown and one plan per goal with a fixed start state of (a) and no interleaving or abandonment, E is simply

$$\begin{bmatrix} 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0 \\ 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0 \\ 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0 \end{bmatrix}. \quad (1)$$

E will be much larger when interleaving and/or abandonment are allowed. The Tn columns correspond to the Tn observations in the trace (in row major order), while each row of E is a binary encoding of an occurrence (Definition 2). In equation 1, the three rows from top correspond to the three occurrences indicated in Figure 1(b) for goals ‘‘TAR’’, ‘‘TAX’’ and ‘‘AXE’’ respectively. Essentially, a row has a 1 for every non-noop trace-cell covered by that occurrence, and 0 elsewhere. We can consider a binary vector, x of length $r(E)$, that stipulates which rows of E (i.e., which occurrences) can yield a solution. In other words, x gives a set of rows of E such that for every column of E there is exactly one row in this set that contains a 1 in that column. In the example corresponding to equation 1, $x = [1, 0, 1]$, where only the ‘‘(noop)’’ columns are not covered. Then the MAPR problem can be posed as the master integer program (MP)

$$\begin{aligned} \max \quad & v^T x \\ \text{s.t.} \quad & E^T x = 1 \\ & x_i \in \{0, 1\} \forall i = 1, \dots, r(E) \end{aligned} \quad (2)$$

where v is the vector of utilities of the occurrences. The restricted master problem (RMP) is given by

$$\begin{aligned} \max \quad & \tilde{v}^T \tilde{x} \\ \text{s.t.} \quad & \tilde{E}^T \tilde{x} = 1 \\ & \tilde{x}_i \in \{0, 1\} \forall i = 1, \dots, r(\tilde{E}) \end{aligned} \quad (3)$$

where \tilde{E} contains a much smaller number of rows (but Tn columns), and \tilde{x} is the decision vector on this restricted basis. We are also required to relax the RMP to be solved as a *linear program*, thus producing a vector \tilde{x} such that potentially $0 < \tilde{x}_k < 1$ for some k . To force these variables to be boolean, we branch on a set of fractional variables, determined as follows: We pick two columns of \tilde{E} , indexed i and j (say), such that $E(k, i) = E(k, j) = 1$ for rows k with $0 < \tilde{x}_k < 1$ in the LP solution. That is, the trace cells corresponding to i and j are both covered by some occurrence(s) with fractional weight in \tilde{x} . This gives us a set of rows K of \tilde{E} such that $0 < \sum_{k \in K: E(k, i) = E(k, j) = 1} \tilde{x}_k < 1$. We form two branches: one with the added restriction that $\sum_{k \in K: E(k, i) = E(k, j) = 1} \tilde{x}_k = 1$, i.e., both trace cells corresponding to i and j must be covered by exactly one row from K . The other branch has the restriction $\sum_{k \in K: E(k, i) = E(k, j) = 1} \tilde{x}_k = 0$, i.e., no row from K can cover those trace cells. These branches not only ensure that the fractional weights are forced to become boolean further down the tree, but also that the resulting problems are separated to minimize overlap/redundancy. Each resulting node in the search tree is a new RMP. Instead of adding the above new constraints to the RMP, equivalent constraints are actually added to the slave problem (SP) in the successor nodes. For the two successors corresponding to the two branches above, we add the constraints $E(k, i) = E(k, j)$ and $E(k, i) + E(k, j) \leq 1$ respectively, for every row k added by column generation in the subtree rooted at this successor. We also remove occurrences that do not satisfy these constraints from the basis in the successors. These steps are standard for covering problems (Wolsey 1998), so we skip the explanations.

Column Generation

The general idea of column generation is that optimal solutions to large LPs can be produced without explicitly including all variables (x , corresponding to the columns of E^T) in the constraint matrix E . In fact, only a small subset will be in the solution and all others that have a non-positive *reduced price* (Wolsey 1998) can be ignored.

When the LP form of the RMP (equation 3) is solved, let the primal and the dual solutions be denoted by (x, μ) . μ may not be dual feasible for the LP version of the master IP (equation 2), but this can be verified by checking for non-positive reduced price

$$\exists k \text{ s.t. } (v_k - \mu E_k^T) \leq 0? \quad (4)$$

Of course, since we do not know E , this cannot be directly solved. Instead this is solved as the *pricing subproblem* (SP) as discussed in the next section.

If no such k is found, then μ is indeed dual feasible and hence the optimal solution of the LP version of equation 3 is also optimal for the LP version of equation 2. Then we branch as stated in the previous section. But if such a k is found, then this new variable is included in the basis, and the corresponding column is added to \tilde{E}^T , and column generation is repeated.

The example of Figure 1 is too impoverished to illustrate the entire process of column generation and branch and

price. In fact, a detailed illustration will require a rather unwieldy example. Instead, we try to provide the intuitions through an analogy: consider the problem of fitting a puzzle base (\mathcal{T}) with jigsaw pieces (analogous to occurrences) that include many unnecessary pieces. While this problem can be posed as equation 2, we allow pieces to be fractionally selected ($0 < x_k < 1$) from a smaller set of pieces (\tilde{E}), producing equation 3. Then, equation 4 (the pricing subproblem, as described in the next section) is analogous to asking for the *best* piece to be added to a partially filled jigsaw base. The associated constraint $E(k, i) = E(k, j)$ requires that the returned piece either covers both areas i and j of the base (i.e., $E(k, i) = E(k, j) = 1$) or covers neither (i.e., $E(k, i) = E(k, j) = 0$). The constraint $E(k, i) + E(k, j) \leq 1$ requires that the returned piece must not cover both areas i and j of the base simultaneously. The incremental filling of the jigsaw base would be controlled by the outer branch and bound process.

The Pricing Subproblem

The key component of the column generation approach that is non-standard and cannot be used off-the-shelf, is the pricing subproblem that leads to expansion of \tilde{E} . In MAPR, the pricing problem can be setup as a separate integer programming problem that returns the maximizing k for equation 4. We create a graph that maps the non-noop cells in \mathcal{T} to matching plan steps in the library, as shown in Figure 2. If a cell c is mapped to step s_p of a plan p then the (dashed) edge is represented by the boolean variable x_{cs_p} . Each cell c is associated with the time t_c when it was executed and the agent a_c that executed it, i.e., $\mathcal{T}(t_c, a_c) = c$.

We first present the constraints of this IP for the cases where activity interleaving are present and absent, followed by our choice of v to give the objective function. In what follows, M is assumed to be a large integer, at least $M > \max\{T, n, \pi_M\}$ where π_M is the maximum number of steps in any plan.

Interleaving Allowed

Each plan step s_p that is not the “START” step is assigned a time τ_{s_p} , giving a pair of constraints for each s_p (except “START”): $\sum_c (t_c - M)x_{cs_p} \leq \tau_{s_p} - M$ and $\sum_c (t_c - 2M)x_{cs_p} \geq \tau_{s_p} - 2M$. There is a boolean variable b_p corresponding to the “START” step of a plan p saying whether it (and hence the plan) is active. Since the program

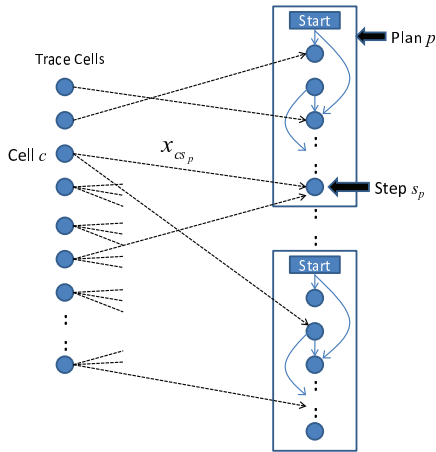


Figure 2: Graph of the pricing problem.

must output one occurrence, $\sum_p b_p = 1$. Then for each step (l_p) for each plan p that has an incoming ordering edge from the “START” step of p (we call these “leader” steps), we have the constraint $\tau_{l_p} \geq (1 - b_p)M + 1$.

We use two variables τ_{\min} and τ_{\max} to locate the start and end times of the occurrence returned. Then for each plan leader step l_p , we have the constraint $\tau_{\min} \leq \tau_{l_p}$, and for each plan step s_p of p there is a constraint $\tau_{s_p} \leq \tau_{\max} + (1 - \sum_c x_{cs_p})2M$.

Each ordering constraint (edge $s'_p \rightarrow s_p$) and each concurrency constraint (edge $s_p \leftrightarrow s'_p$) in each plan p are respectively encoded as $\tau_{s_p} > \tau_{s'_p}$ and $\tau_{s_p} = \tau_{s'_p}$. Each role constraint (edge $s_p \Leftrightarrow s'_p$) in each plan p is encoded as

$$\sum_c (a_c - M)x_{cs_p} + 2M \geq \sum_c (a_c + M)x_{cs'_p}$$

$$\sum_c (a_c + M)x_{cs_p} \leq \sum_c (a_c - M)x_{cs'_p} + 2M$$

Finally, the constraint $\sum_{s_p, p} x_{cs_p} \leq 1$ for each c ensures that no more than one outgoing edge from c can be active. Similarly, the constraint $\sum_c x_{cs_p} \leq 1$ for each step s_p of each plan p ensures that no more than one cell can map to s_p (at most one incoming active edge to s_p).

In the blocks word domain, the fact that an agent holding a block X (due to some action θ) cannot execute any non-noop action unless it has executed an action θ' of the form (stack X^*) or (put-down X), or has abandoned the plan (in which case it is assumed to have executed an invisible “reset” action) cannot be adequately captured by the role and ordering constraints alone. For this purpose, we use the following set of constraints only for the blocks word domain: for each ordered pair $\theta \rightarrow \theta'$ described above in each plan p , for each cell c that maps to θ , pick c' – the earliest non-noop cell corresponding to agent a_c after time t_c ; then

$$\sum_{s_p} \sum_{c' \in \Theta} x_{c's_p} \leq (1 + x_{c'\theta'} - x_{c\theta})M$$

where Θ is the set of all (non-noop) cells corresponding to agent a_c after time t_c (therefore, Θ includes c'). Here we are enforcing the convention that if c' has no mapping to θ' (in Figure 2) then $x_{c'\theta'} = 0$ instead of being undefined. There are $O(\pi_M^2 L + Tn)$ constraints and $O(\pi_M L T n)$ variables, where $L = |\mathcal{L}|$.

Interleaving Not Allowed

Additional constraints are needed when the agents are not interleaving their plan steps. To prevent activity interleaving we use the following pair of constraints for each c

$$\tau_{\min} - t_c + (t_c + M) \sum_{s_p, p} x_{cs_p} + M(2 + k_c - 2\alpha_i) > 0$$

$$\tau_{\max} - t_c + (t_c - M) \sum_{s_p, p} x_{cs_p} - M(1 + k_c - 2\alpha_i) < 0$$

where $i = a_c$, the agent corresponding to cell c . These constraints ensure that if agent $i = a_c$ is active (i.e., $\alpha_i = 1$)

but c is inactive, then $t_c \notin [\tau_{\min}, \tau_{\max}]$. The boolean variable k_c needs no additional constraint, but for the boolean variable α_i we have a pair of constraints for each agent i : $\sum_{a_c=i, s_p, p} x_{cs_p} \leq \alpha_i M$ and $\sum_{a_c=i, s_p, p} x_{cs_p} \geq \alpha_i$. Thus, there are an additional $O(Tn)$ constraints and n variables compared to the interleaved case.

Objective Function

To generate the utility of occurrence o^k (given as v_k in equation 4), we represent its posterior likelihood as $P(o^k|X^{o^k}, \pi) \cdot P(\pi|X^{o^k}) \cdot P(X^{o^k})$ where $P(X^{o^k})$ stands for the likelihood of agents in X^{o^k} forming a team at time $t_{\min}^{o^k}$, $P(\pi|X^{o^k})$ stands for the likelihood of that team selecting plan π , and $P(o^k|X^{o^k}, \pi)$ is the likelihood of o^k emerging as the observational outcome of that team executing that plan. For this paper, we assume a very simple yet reasonable generative model of these likelihoods: $P(X^{o^k}) \propto \exp(-\beta_1|X^{o^k}|)$ thus making smaller teams likelier than larger teams, $P(\pi|X^{o^k}) \propto \exp(-\beta_2(|\pi| - |X^{o^k}|))$ thus making smaller teams prefer smaller plans ($|\pi|$ being the number of steps in π), and $P(o^k|X^{o^k}, \pi) \propto \exp(-\beta_3(|\pi| - |o^k|))$ for no interleaving (thus preferring plans closer to completion than further) but $P(o^k|X^{o^k}, \pi) \propto \exp(-\beta_3(|\pi| - |o^k|) - \beta_4(t_{\max}^{o^k} - t_{\min}^{o^k}))$ with interleaving thus also preferring plan execution during shorter intervals over longer/largely separated intervals. Setting v_k to the posterior log-likelihood, we get

$$v_k = (\beta_2 - \beta_1)|X^{o^k}| - (\beta_2 + \beta_3)|\pi| + \beta_3|o^k| - \beta_4(t_{\max}^{o^k} - t_{\min}^{o^k})$$

where $\beta_4 = 0$ for the case of non-interleaved explanations. Note that setting $f(\{o^1, \dots, o^z\}) = \sum_i v_i$, gives us the MAP explanation under the above probability model and the assumption of independence of occurrence likelihoods.

Experimental Results

We selected two domains for experimentation: BLOCKS WORD and INTRUSION DETECTION (Geib and Goldman 2002), for both of which the data were adopted from (Ramirez and Geffner 2010), hence the goals were all conjunctive. We selected the 5 goal words ‘‘STAR’’, ‘‘STACK’’, ‘‘RASH’’, ‘‘TRASH’’ and ‘‘CRASH’’, and 2 possible start states (as shown in Figure 4) for BLOCKS WORD. With a single plan per start-goal configuration, we had 10 plan graphs in the library with an average of 15 steps per plan. For INTRUSION DETECTION, we randomly selected 5 goals from (Ramirez and Geffner 2010) and assumed no role or concurrency constraints. Note that this domain is more abstract and has no ground states, unlike BLOCKS WORD. With one plan per conjunctive goal, we had 5 plan graphs in the plan library with an average of 16 steps per plan. We modified the domain and problem PDDL descriptions to accommodate a variable number of agents for plan execution, then used a partial order planner to populate \mathcal{L} (e.g., Figure 1(c)) given the goals, and finally allowed distributed execution of randomly selected plans by random (dynamic)

teams to generate the traces ¹.

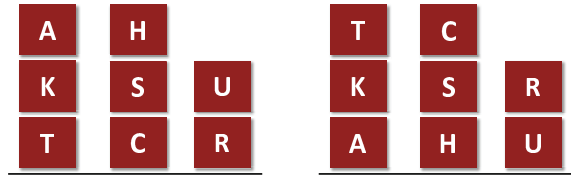


Figure 4: Start states for BLOCKS WORD experiments.

We compare our results with the hypothesis pruning approach of (Banerjee, Kraemer, and Lyle 2010) (labeled ‘‘Algorithm X’’ in the plots and ‘‘ALGX’’ in the table), since it is based on a model closest to ours. We used IBM ILOG CPLEX (academic version) for the branch and price implementation (labeled ‘‘BNP’’ in the table) but disabled its multi-core utilization for fair comparison with ‘‘Algorithm X’’. The experiments were run on a cluster with 12 Dell M610 Blades, each having 8 Intel Xeon 3.00GHz processors with 12Gb RAM shared amongst them, with a cutoff time of 12h. We set $\beta_1 = \beta_3 = \beta_4 = 1, \beta_2 = 2$. Figure 3 shows the plot of runtimes, proving the (relative) scalability of the new perspective to search. The times reported for ‘‘ALGX’’ includes the occurrence generation time, since ‘‘Algorithm X’’ needs all occurrences to be generated prior to search. Table shows the relative number of occurrences seen by the two search approaches, supporting the above conclusion, and also shows the percentage of instances that were cutoff by the time limit. Results were not available for ‘‘N/A’’ because the runs were too expensive. We kept T fixed at 15, and varied n since the latter is the key to intractability. All results are averaged over 30 instances including the cutoff instances, which explains why some plots flatten out close to the cutoff time. Thus except for BLOCKS WORD with no interleaving, we see a speedup of 100 times or better for our proposed hypotheses growth approach, compared to hypotheses pruning.

Conclusions and Future Work

We have presented an adaptation of a powerful OR technique, branch and price, for solving instances of large MAPR problems that accommodate activity interleaving and plan abandonment. The underlying principle can be thought of as a hypotheses growth approach, as opposed to the prevailing hypotheses pruning approach. Our experiments show that the new approach is significantly more scalable than a previous approach. Our formulation of MAPR is geared toward bringing other mature OR methodologies to bear upon MAPR in the future, and envisaged to have a similar impact as mature SAT-solvers had on planning.

Acknowledgments: We are grateful to the anonymous reviewers, as well as to Gal Kaminka and Gita Sukthankar for helpful comments and suggestions. This work was supported in part by a start-up grant from the University of Southern Mississippi.

¹A part of the trace generator utility is available at <http://www.cs.usm.edu/~banerjee/TraceGen>

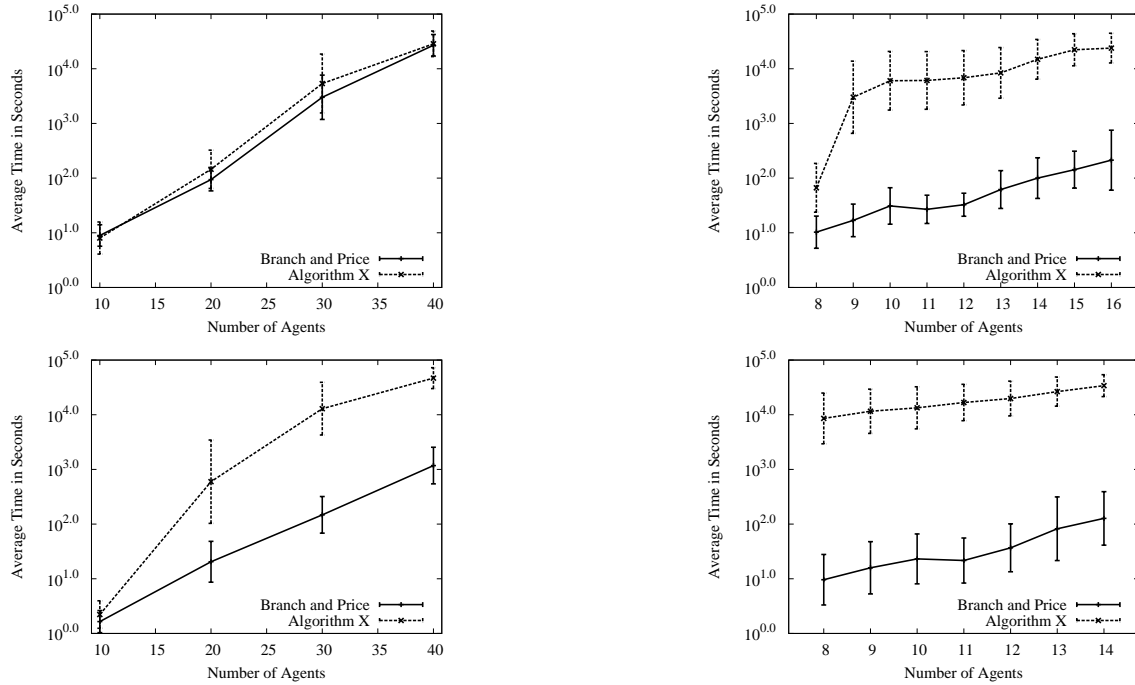


Figure 3: Top: Multi-agent BLOCKS WORD with no interleaving (left), and with interleaving (right). Bottom: Same for Multi-agent INTRUSION DETECTION

n	INTRUSION DETECTION								BLOCKS WORD							
	Interleaving				No Interleaving				Interleaving				No Interleaving			
	% cutoff		# occs $\times 10^3$		% cutoff		# occs $\times 10^3$		% cutoff		# occs $\times 10^3$		% cutoff		# occs $\times 10^3$	
	ALGX	BNP	ALGX	BNP	ALGX	BNP	ALGX	BNP	ALGX	BNP	ALGX	BNP	ALGX	BNP	ALGX	BNP
8	16.7	0.0	479.9	0.1	0.0	0.0	0.2	0.0	0.0	0.0	61.2	0.0	0.0	0.0	0.3	0.0
10	26.7	0.0	860.9	0.1	0.0	0.0	0.2	0.0	13.3	0.0	139.9	0.1	0.0	0.0	0.5	0.0
12	40.0	0.0	2270.0	0.2	0.0	0.0	0.3	0.0	10.0	0.0	225.5	0.1	0.0	0.0	0.8	0.0
14	70.0	0.0	5060.6	0.3	0.0	0.0	0.5	0.0	30.0	0.0	540.5	0.1	0.0	0.0	1.0	0.0
16	N/A	0.0	N/A	0.4	0.0	0.0	0.8	0.0	46.7	0.0	871.5	0.2	0.0	0.0	1.6	0.0
20	N/A	0.0	N/A	0.8	0.0	0.0	2.3	0.0	N/A	0.0	N/A	0.4	0.0	0.0	2.9	0.0
30	N/A	0.0	N/A	2.1	20.0	0.0	5.2	0.1	N/A	6.7	N/A	1.7	10.0	0.0	17.3	0.2
40	N/A	56.7	N/A	2.8	80.0	0.0	22.1	0.2	N/A	86.7	N/A	2.0	50.0	33.3	54.0	0.4

References

- Banerjee, B.; Kraemer, L.; and Lyle, J. 2010. Multi-agent plan recognition: Formalization and algorithms. In *Proceedings of AAI-10*, 1059–1064.
- 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.
- Geib, C., and Goldman, R. 2002. Requirements for plan recognition in network security systems. In *Proceedings of Int. Symp. on Recent Advances in Intrusion Detection*.
- Geib, C. W.; Maraist, J.; and Goldman, R. P. 2008. A new probabilistic plan recognition algorithm based on string rewriting. In *Proc. ICAPS*.
- Ghallab, M.; Nau, D.; and Traverso, P. 2004. *Automated Planning: Theory and Practice*. Morgan Kaufmann Publishers.
- Kaminka, G., and Bowling, M. 2002. Towards robust teams with many agents. In *Proc. AAMAS-02*.
- Kautz, H. A., and Allen, J. F. 1986. Generalized plan recognition. In *Proc. AAAI*.
- Ramirez, M., and Geffner, H. 2010. Probabilistic plan recognition using off-the-shelf classical planners. In *Proceedings of AAI-10*, 1121–1126.
- Sadilek, A., and Kautz, H. 2010. Recognizing multi-agent activities from gps data. In *Proceedings of AAI-10*, 1134–1139.
- Sukthankar, G., and Sycara, K. 2008. Hypothesis pruning and ranking for large plan recognition problems. In *Proc. of AAAI*.
- Wolsey, L. A. 1998. *Integer Programming*. Wiley.