Multi-Agent Approach to Plan Recognition

Ratul R. Pradhan[1], **Todd W. Neller**[1], **Richard G. Freedman**[2]

[1] Gettysburg College
[2] SIFT
pradra01@gettysburg.edu, tneller@gettysburg.edu, rfreedman@sift.net

Abstract

Contributions to the plan recognition problem have assumed a single agent acting in the world, i.e. all the observations in the observation sequence belong to a single entity. However, there may be cases where multiple agents may be acting on the same world for which the existing work cannot effectively sole the issue. This paper introduces a method to do so, through converting observation as actions into observations over fluents, followed by seperation of each agent's action by a unique identifier. (Further description required).

1 Introduction

Plan Recognition is the process of recognizing an entity's goal via observing it's behavior. To put it simply, a sequence of observations and actions are be given with a possible number of goals, to which predictions are made as to what may be the entity's actual goal. This may also be considered be Goal Recognition we essentially aim to find the goal through analysing the observation sequence. Using a domain theory provides this approach to be more generic, accessible, and scalable compared to a plan library.

Advancements in Plan recognition have been made independent to planning. This paper utilizes Ramírez and Geffner's (Ramirez and Geffner 2009) work on plan recognition over a *domain theory* and a possible set *G* of goal, where optimality of the agents will be assumed, i.e. they represent the behaviors of a perfectly rational entity, where the enity is assumed to take only optimal actions. It should be noted that this work, while creating the basis for a majority of the work that would follow after, had a number of limitations. In (Ramırez and Geffner 2010), they devised a method of generating posterior probabilities rather than boolean judgements, addressing cases where the goals could be ranked through most likely, to least. Another advancement was the effective handling of unreliable observations (Sohrabi, Riabov, and Udrea 2016), i.e. noisy (observations were never added to the state by any actions) or missing (observations were part of the domain but not part of the observation sequence) using observations over fluents. Masters and Sardina (Masters and Sardiña 2017) focus on recognition specifically for the path-planning domain, highlighting how the probability distribution that ranks goals can be obtained without referencing any observations except it's most current location, using a path-planning approach of including waypoints, allowing retrieval of in constant time and the possibility of pre-computing the results.

All of these advancements make the assumption of a single-agent world, i.e. the existence of a sole agent acting in a static world where the domain consists of only one agent's actions, observations, and goals. However, interactive systems and multi-agent systems have an ever-increasing presence in research and society at large today where it is necessary to be able to derive and compute multi-agent plan recognition problems as traditional plan recognition methods may fail. The current implementations of recognition as planning does not allow for the presence of more than one entity This paper aims to tackle this limitation by formulating a general method of multi-agent inclusive plan and goal recognition.

2 Background

Plan Recognition over Domain Theory

This section will go over the plan/goal recognition problem over a *domain theory* (Ramirez and Geffner 2009; Ramırez and Geffner 2010). Their method includes the functionality of generating new plans that incorporate the observations rather than using the pre-existing plans, thereby creating two unique plans for each goal. This method also using STRIPS notations.

A planning domain *is a tuple D = (F, A) where F is a set of fluents and A is the set of the actions a, where each action a contains a precondition list Pre(a), add list Add(a), and delete list Del(a), which are all subsets of F.*

Definition 1 *Example* test *something.*

A planning problem *is a tuple P = (F, A, I, $\mathcal{G}$) where F and A contain the planning domain, where $I \subseteq F$, is the initial state (fluent), and $G \subseteq F$ is the goal state (set of fluents). π is the solution plan from I to G where $\pi = [a_0, a_1, ...a_n]$. π is said to be optimal if cost(π) = Σ cost(a_i)*

A plan recognition problem *is a tuple R = (F, A, I, G, Prob) where (F, A, I) are as defined above, $G \subseteq F$ is a set of possible goals and Prob is the posterior probability distribution over G*

The cost *c(G, O)* and *c'(G, O)* are the costs for the cheapest (optimal) plan to the goal with all observed actions taken and the cheapest (optimal) plan to the goal with at least one

observation missing. The cost difference between these two is then calculated as:

$$costdiff(G, O) = c(G, O) - c'(G, O) \quad (1)$$

Through comparing all the cost differences for all G ⊂ G, a new probability distribution is formed where *the probability of a certain goal being the solution is higher, the lower the cost difference for that goal* . Assuming a Boltzman distribution, they define the following

Add rest

Observations over Fluents

Sohrabi, Riabov and Udrea's paper (Sohrabi, Riabov, and Udrea 2016) extends the above defined work with the possibility of defining observations as fluents, which is aimed to better address missing and noisy observations. They argue that in many applications, the actual actions of an agent may not be observable, instead, their effects through change in states are observable.

Noisy referring to observations that are never added to the state by any actions a simpler way to describe it: an incorrect observation and missing referring to observations that are part of the domain everything in the world is part of the domain; I think you mean plan? but not part of the observation sequence.

Each observation is defined as a logical expression over the set of fluents. Furthermore, observations over actions may also be dealt with in this version through assigning a unique fluent per action that is added by that specific action. The technique is excellent here, but the reader is hoping to understand the background, which goes beyond what is done to how/why is it done?

The new plan recognition problem incorporates the existing actions with a set of "discard" and "explain" actions for each observation in the observation sequence, allowing for the observations to be explained by the actions, or discarded if they cannot be explained (allowing noisy observations to be skipped). This order of observations through this separation is maintained by a special predicate, l_O comma goes here which is set true if the o^{th} observation can be explained or discarded. Furthermore, o_0 what is o_0? I do not recognize this change to the observation sequence is added for the initial state with l_{O_0} set as true, and another special predicate *"done"* is added for each G ∈ G ensuring the search space only considers plans that meet at least one of the goals. b_1 *and* b_2 *are coefficients for missing and noisy observations respectively which assigns weights for different goals.*

Hence, ***a planning problem*** *P' = (F', A', I', G') is created where:*

- *F' = F* $\cup$ $\{done, l_{o_0}\} \cup \{l_{o_i} | o_i \in O\}$
- *add the rest*

This should go much earlier because it is a prerequisite for understanding any of the related works included in this section.

Cost Based Goal Recognition

Masters and Sardina's work shows a method finding a probability distribution that ranks candidate goals in the same order is this guaranteed when the special case occurs? Beware false advertising. without referencing any observations other than the agent's current location, essentially making their technique observation independent. If the current location is an observation, then it is still observation-dependent. I think you are referring to independence of the rest the observation sequence. Also, Masters and Sardena's method only applies to path planning tasks rather than any STRIPS-represented task. This allows the distribution to be calculated in constant time this is false because the calculation still runs a planner to solve STRIPS, and Bylander (1994) proves that solving STRIPS problems is PSPACE-Complete (frighteningly complex... this complexity makes NP look simple, and NP is much more complex than constant-time). and provides the possibility of a pre-computing step that could have one of the plans on hand. It is easy to get excited about a method you like, but you cannot claim that it does more than it actually does. This was what the neural networks community did long ago before deep learning existed, and the lies caught up to them and led to funding agencies hating AI research for many years (the AI Winter).

A path-planning goal recognition (PPGR) problem, *according to their work, is a tuple P = (D = (D, G, s, O, Prob) where Is the "(D =" part supposed to be here?*

- *D = (N, E, c) is the domain with non-empty set of fluents N, set of actions/edges the reader has no idea why actions would be called edges because you neither explained path planning nor showed how planning domains are graphs E, and a function c that returns the non-negative cost of traversing each edge. Waypoints are further specified as nodes that must be visited, embeded to preserve order. I recommend you give a brief explanation of path planning so that this definition makes sense.*
- *G* $\subseteq$ *N is a set of possible goals*
- *s* $\in$ *N is the start location*
- *O* = $o_1, ..., o_k$ *where k* $\geq$ *0 and* $o_i \geq N$ *What does it mean to be greater than or equal to a set? Did you mean set containment/in* $\in$*?, is the sequence of observations where* $o_1 \neq s$
- *Prob represents the prior probabilities of the goals*

A reworking for the simple cost difference gives:

$$costdiff(s, G, O) = c(s, O, G) - c(s, G)$$

This should be either an equation or displaymath environment: \begin{equation or displaymath}. If you choose displaymath, then you can also use double the math mode symbol $$...$$ as a shortcut.

Instead of finding c'(G, O), i.e. the cost for the cheapest plan with at least one observation missing, the optimal path cost is deducted. Correct technique, but again without explanation. Readers in the background section want to know what is going on without having to read the other paper (unless they get interested in the work), and that means you must explain it briefly.

3 Methods

Modifications for incorporating multiple agents

Delete this! Describe your procedure and experiments. Make sure the assumptions, steps, equations, models, and approaches are clearly explained. Others should be able to follow this section and replicate your research.

With the aforementioned background in order, we now move towards creating a planning problem supporting multiple agents: Use the definition environment here to get the formatting and consistent numbering without all the hassle.

Definition 1. The new planning domain *is a tuple D = (F, A) where A now contains a special predicate y_{a_i} that uniquely identifies the agent and all of it's corresponding actions, such that $y_{a_i} = y_i$ where $\{y_i | y_i \in Y\}$,*
$Y = \{y_1, ...y_n\}$ for each unique agent acting in the world. Several questions: (1) is y really a predicate? (2) Your equivalence notation needs another round because $y_{a_i} = y_i$ means that $a_i = i$, and that is comparing actions to numbers if I understand the rest your representation choices.

Definition 2. The new plan recognition problem *is a tuple R = (F, A, I, G, Prob, Y), with the addition of Y, where Y represents the number of agents acting in the world Y = $\{y_1, y_2, ...y_n\}$.*

With these new additions, we can essentially check which agent performs which action by matching the Y values of the actions with the set Y on the plan recognition problem.

This method will be assuming observations as fluents, rather than actions. Hence, to convert the plan recognition problem into a planning problem through action costs, we modify Sohrabi, Riabov and Udrea's method of incorporating existing actions with a "discard" and "explained" set, to rather incorporating them into sets of y_n, i.e. a different set of actions for each agent, thereby separating each agent's actions sequence. This run-on sentence needs to be broken down and written more clearly.

We utilize $l_{o_{init}}$ should this be $init$? and set it to true if the observation is incorporated into the sets of y_n, i.e. can be explained through at least one of the agent's action, with dummy initial o_0 and l_{o_0} for the initial observations and predicate *done* to define the end fluent. What is the end fluent—the last observation in the sequence?

Hence, $l_{o_{init}}$ essentially tells us that the current observation is the i^{th} observation in the observation sequence. Again, should this be $init$ or i?

Furthermore, to ensure we deal with the observations over actions, we add a unique fluent x_i, $\{x_i | x_i \in X\}$ for Why did you put the break here for the figures? This destroys your single paragraph. If it looked like this is where the images should go, then it might help to know that LaTeX guesses figure placement in order to avoid interfering with paragraphs. It will put it at the top of the current or next page, unless they are full. Then, LaTeX finds the best page available during its compilation step.

each of the specific actions to convert into observations over fluents. If you already explained this technique in your background, then you can refer back to it. Otherwise, please define X here because you never used that variable name before. Finally, b_1 is a coefficient assigning weights. What is b_1's value? If arbitrary or does not matter, then say so and explain why. Thus, we get:

Definition 3. The tuple R = (F, A, I, G, Prob, Y) is modified *such that for each $a_i \in A$, we add a new unique fluent x_i into its add list Add(a).* What term are you defining here?
Therefore add a comma here $F = F \cup X$, where $\{x_i | x \in X\}$ $x_i \in X$? Also, one of the F's should be renamed because it is an expanded set (rather than the same set). such that $X = \{x_1, x_2, ...x_n\}$ for each action a_i in A. This is very confusing to read and might be circular. You define X based on X, and then you rewrite X again and relate it to A.

Definition 4. A new planning problem *P' = (F', A', I', G') is created from the new plan recognition problem R = (F, A, I , G, Prob, Y), where :*

- $F' = F \cup \{done\} \cup \{l_{o_i} | o_i \in O\} \cup \{l_{o_{init}}\}$
- $I' = I \cup \{l_{o_{init}}\}$
- $G' = \{done\} \cup \{l_{o_m}\}$, *where o_m is the last observation*
- $A' = A_{orig} \cup A_{goal} \cup A_{y_i}, \{y_i | y \in Y\}$ $y_i \in Y$?
  - $A_{orig} = \{h_a | a \in A, COST(h_a) = COST(a) + b_1 \times |\{o_i | o_i \in ADD(a) \wedge o_i \in T \wedge o_i \notin O\}|$, *where $T \subseteq F$ is the set of observable fluents, i.e any fluent that has the potential to be observed.*
  - $A_{goal} = \{h_g | g \in \mathcal{G}, COST(h_g) = 0, PRE(h_g) = \{g\}, ADD(h_g) = \{done\}\}$ *Be careful about changing your notation in the middle of a paper; you did not call these functions COST, ADD, PRE, etc. before.*
  - *for each y_i in Y, we create an action set A_{y_i} that adds actions that consist of the matching y_{a_i} predicate.*
    $A_{y_i} = \{h_{o_i} | o_i \in O, COST(h_{o_i}) = 0,$
    $PRE(h_{o_i}) = \{o_i, lo_{i-1}\}, ADD(h_{o_i}) = \{l_{o_i}\},$
    $DEL(h_{o_i}) = \{lo_{i-1}\}\}$

4 Illustrated Example

To demonstrate the compilation process, we will go through an illustrated example of each step in order to verify Is 'verify' the correct word here? each step of the process. We create a simple –two blocks– (remove this) version of blocks world with two blocks. We assume there are two agents working in the world. We also add $\{xi | xi \in X\}$ to the set of fluents How is this different than just adding X to the set of fluents? More importantly, is 'include' more appropriate than 'add' because these are sets rather than numbers?, and add the respective x_i to each action. In order optimal space usage this is missing a word or two and streamline the example, we take the liberty of assigning the names for the predicates x_i as seen fit with blocks world, i.e. each x_i is followed by parenthesis enclosing blocks that are effected 'affect' is a verb, 'effect' is a noun by that action and is solely a stylistic choice, as is the case with the representation of actions definitions being lifted with the use of ppdl-esque PDDL-esque? variables $(?x)$ to define every action in the sub-category of

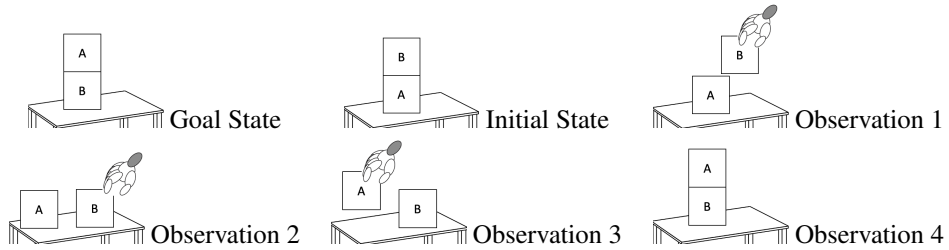

Figure 1: Version 1 of figure layout, which loses the ability to label each image separately. It is intuitive code to read, though.

Figure 2: Goal State

Figure 3: Initial State

Figure 4: Observation 1

Figure 5: Observation 2

Figure 6: Observation 3

Figure 7: Observation 4

Figure 8: Version 2 of figure layout, which gains the ability to label each image separately. It is not easy code to read, though.

that particular action. First of all, that last sentence is a run-on longer than any other run-on you wrote before. Second, your reader has no idea what $?x$ is, even if it is related to PDDL. Keep in mind that no one else was in our semester-long meetings, and you cannot assume anything is obvious just because we talked about it a lot.

Domain and Planning Problem

- **Fluents**:
  ON(A, B), ON(A, Table), ON(B, Table), ON(B, A), HAND(A), HAND(B), HAND(Null), $\cup \{xi|xi \in X\}$ Not required, but I would give you brownie points for explaining what each of these predicates means so that readers can have context. On the other hand, what is the X-related set doing here? (I figured it out in your example; we totally need to explain those are the 'action blah was performed' predicates. It would also help to include their parameters like you did for all the other predicates.
- **Actions**:
  UNSTACK(A, B), UNSTACK(B, A), STACK(A, B), STACK(B, A), UNSTACK(A, Table), UNSTACK(B, Table), STACK(A, Table), STACK(B, Table)
  Each of the actions can be carried out by both agents.
- **Observation Sequence**:
  STACK(A, Table, y_1) , STACK(A, B, y_2) You should explain somewhere that you can use your agent IDs as a parameter for an action. It is typical practice for those using multiple agents, but many people do not use multiple agents and would not know this.
- **Initial State**:
  ON (B, A), ON(A, Table)
- **Goal**:
  Goal for agent y_1, $g_1 = \{$ON(A, B), ON(B, Table) $\}$
  Goal for agent y_2, $g_2 = \{$ON(A, B), $\neg$ ON(B, Table) $\}$

Action Definitions

Each action contains the following lists and predicates What are the square brackets for? You never used these in the past.

- **UNSTACK(?x, ?y)**
  - *Pre[]*= [ON(?x, ?y)]
  - *Add[]*= [HAND(?x), x_i(?x)]
  - *Delete[]*= [ON(?x, ?y)]
- **STACK(?x, ?y)**
  - *Pre[]*= [HAND(?x)]
  - *Add[]*= [ON(?x, ?y), x_i(?x, ?y)]
  - *Delete[]*= [(HAND(?x)]

Verification of Domain and Plan

The figure presented above You can take advantage of your label! Say something like Figure$\sim$ \ref{label name}. is an example of an optimal solution to a simple single agent blocks world problem. We use this to verify that the domain and planning problem –still– (remove) works for single agents and is solvable through the following sequence:

Figure 3 $\rightarrow$ *UNSTACK(B, A)* $\rightarrow$ *Figure 4* $\rightarrow$ *STACK(B, TABLE)* $\rightarrow$ *Figure 5* $\rightarrow$ *UNSTACK(A, Table)* $\rightarrow$ *Figure 6* $\rightarrow$ *STACK(A, B)* $\rightarrow$ *Figure 7*

Thus satisfying the goal state. It should be noted that observation sequence in this case may be assumed as either the actions performed, or the fluents described in the figures as the compilation process can handle both as input. This is an excellent point to bring up; bravo!

However, to solve the multi-agent problem using the method specified above, we make the assumption that the observation sequence in the planning problem are actions, that we then convert into fluents, as we require the predicate y_{a_i} to separate the actions of each agent. Hence, we require observations as actions.

To phrase it differently, we utilize Sohrabi's include the other authors or 'et al.' method of creating unique fluents for

actions to create fluents out of the action-based observation sequence.

Compilation

In this section, we focus on the effects of *Def 4*, You can also create labels in the definition environments to use in more \ref commands. and through example, understand the method of distinguishing each agent's action, thereby allowing us to follow multiple agents in a world. The following demonstration will be detailed but concise, where readers are to make inferences from the described domain and planning problem above.

Following Definition 4, we get:

- $F' = F \cup \{done\} \cup \{l_{x_1(A,Table,y_1)}\} \cup \{l_{x_2(A,Table,y_2)}\} \cup \{l_{o_{init}}\}$
- $I' = I \cup \{l_{o_{init}}\}$
- $G' = \{done\} \cup \{l_{x_n}\}$ where x_n is the last observation.
- $A' = A_{orig} \cup A_{goal} \cup A_{y_1} \cup A_{y_2}$

  To stay concise, we will look at one action and it's components for each category?, and infer similar compilation for the rest. Perfectly phrased!

  - $A_{orig} = \{\text{UNSTACK}(A, B, y_1) | \; Cost_{UNSTACK_{(A,B,y_1)}} = 1 + b_1 \times 0\}$ the \textnormal{text} command in math mode is useful to avoid this mess of entering-and-exiting math mode.

    The example action's *Add()* list contains 2 fluents. Since the blocks world example observes all fluents that are observable, we can deduce that the set of T consists of all fluents. However, all our observations are part of the sequence, thereby not fulfilling every $\wedge$ condition leading to 0. This essentially shows that we have no missing observations, hence our cost for every action remains that same.
  - $A_{goal} = \{h_{\{ON(A,B),ON(B,Table\}} |$
    $Cost_{h_{\{ON(A,B),ON(B,Table\}}} = 0,$
    $Pre() = \{\text{ON(A, B), ON(B, Table)}\}$
    $Add() = \{\text{done}\}\}$
  - $A_{y_1} = \{h_{x_1(A,Table)} |$
    $Cost_{h_{x_1(A,Table)}} = 0$
    $Pre() = \{x_1(A, Table), l_{o_{init}}\}$
    $Add() = \{l_{x_1(A,Table)}\}$
    $Delete() = \{l_{o_{init}}\}\}$
  - $A_{y_2} = \{h_{x_2(A,B)} |$
    $Cost_{h_{x_2(A,B)}} = 0$
    $Pre() = \{x_2(A, B), l_{x_1(A,Table)}\}$
    $Add() = \{l_{x_2(A,B)}\}$
    $Delete() = \{l_{x_1(A,Table)}\}\}$

For future reference, this formatting works just fine, but you might like the align environment as a way to control it more: \begin{align}.

Application of Cost Based Goal Recognition

Now, through this separation of the observations into individual agent's actions, . Does something go between this comma and period?

$costdiff\,(s, G, O) = c(s, O, G) - c(s, G)\mathcal{G}$

5 Results

Delete this! Provide the outputs that you got after performing the steps you described in the Methods section above (Section 3). This should be objective, providing numbers and plots without any opinions or interpretations. Are the hypotheses correct, incorrect, or inconclusive?

6 Discussion

Delete this! Present interpretations of the results (Section 5) here. These may include opinions about what the results mean, why your hypotheses were (in)correct based on inspecting the results, and other interesting patterns or trends you notice during and/or after the experiments (Shvo and McIlraith 2020).

Did you read the Shvo and McIlraith paper? We never talked about it, but we should if you read it.

7 Conclusion

Delete this! Summarize the entire paper, which covers all the sections (unlike the abstract). Also include future work, which lists follow-up ideas and research questions. These are not commitments, but inspirations from the presented research—either you or some other reader can pursue those projects. Complete the story that began in the introduction (Section 1).

Acknowledgments

Delete this! List people to thank who helped with the research or paper, but are not authors. Do not include this section in the review copy (could give away author identities).

References

Masters, P.; and Sardiña, S. 2017. Cost-Based Goal Recognition for Path-Planning. In Larson, K.; Winikoff, M.; Das, S.; and Durfee, E. H., eds., *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2017, São Paulo, Brazil, May 8-12, 2017*, 750–758. ACM.

Ramirez, M.; and Geffner, H. 2009. Plan Recognition as Planning. 1778–1783.

Ramırez, M.; and Geffner, H. 2010. Probabilistic plan recognition using off-the-shelf classical planners. In *Proceedings of the Conference of the Association for the Advancement of Artificial Intelligence (AAAI 2010)*, 1121–1126. Citeseer.

Shvo, M.; and McIlraith, S. A. 2020. Active Goal Recognition. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(06): 9957–9966.

Sohrabi, S.; Riabov, A. V.; and Udrea, O. 2016. Plan Recognition as Planning Revisited. In Kambhampati, S., ed., *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, 3258–3264. IJCAI/AAAI Press.