Multi-Agent Plan Recognition as Planning (MAPRAP)

Chris Argenta and Jon Doyle
North Carolina State University, Raliegh NC, U.S.A.

Keywords: Multi-Agent Systems, Plan Recognition.

Abstract: Multi-agent Plan Recognition (MPAR) infers the goals of teams of agents from their observed actions. Recognizing action sequences spread over an unknown team composition significantly increases the complexity of creating a priori plan libraries. A key challenge in MPAR is effectively pruning the large search space of goal to team assignments and agent to team compositions. In this paper, we describe discrete Multi-agent Plan Recognition as Planning (MAPRAP), which extends Ramirez and Geffner's Plan Recognition as Planning (PRAP) approach into multi-agent domains. MAPRAP uses a planning domain (rather than a library) and synthesizes plans to achieve hypothesized goals with additional requirements for suspected team composition and previous observations. Recognition is accomplished by comparing the planning results and identifying feasible combinations of plans and teams being observed. We establish a performance profile for discrete MAPRAP in a multi-agent blocks-world domain. We vary the number of teams, agent counts, and goal sizes and measured accuracy, precision, and recall at each time step. We also compare two pruning strategies for discrete MAPRAP that dampen the explosion of hypotheses. More aggressive pruning recognizes multi-agent scenarios with an average of 1.05 plans synthesized per goal per time step as opposed to 0.56 for single agent scenarios demonstrating feasibility of MAPRAP and benchmarking for future improvements.

1 INTRODUCTION

Recognizing the plans of multi-agent teams from remote observation of their actions enables people and intelligent systems to make sense of complex behaviors, identify collaborations pursuing goals of interest, and better predict the future actions and states of the world. The process of identifying cooperative plans and the composition of teams pursuing them is called Multi-agent Plan Recognition (MPAR) (Sukthankar et. at., 2014). MAPR attempts to identify plans underlying the observable actions of a collection of agents, which are organized into teams that share a common goal and perform coordinated planning. An online recognizer observes actions conducted by agents over time and at each time step infers a set of interpretations for the scene. These interpretations including which agents are working together as a team, what goal(s) each team is pursuing, and how they may intend to achieve their goal(s) in the form of a multi-agent plan.

In this paper, we describe two variants of discrete Multi-agent Plan Recognition as Planning (MAPRAP), which extends Ramirez and Geffner's (2009, 2010) Plan Recognition as Planning (PRAP) approach into multi-agent domains. We outline the core challenges and present our approach for evaluation. Finally, we give results for a version of the well-established Blocks World domain (e.g., Ramirez and Geffner, 2009; Zhou et al., 2012; Banerjee et al., 2010). These results serve as a baseline for on-going research exploring alternative strategies (e.g., probabilistic), application to other benchmark multi-agent domain, and comparing recognition performance under less ideal conditions (e.g., missing observations or competing teams).

MAPRAP is intended as a general plan recognition technique, meaning that it can be applied to any domain provided the necessary inputs and, until fully general, stated assumptions are met. Our focus on disallowing prior domain knowledge follows the lead of General Game Playing (GPP) (Genersereth and Love, 2005) and International Planning Competition (IPC) communities. In MAPRAP, the planning domain is based on Plan Domain Description Language (PDDL) (McDermott et al., 1998) annotated for multiple agents (similar to

MA-PDDL (Kovacs, 2012) conversion via (Muise et al., 2014)). This domain includes a complete initial state, list of agents, list of potential goals, and action model.

In contrast, most plan recognition techniques match observables to patterns within a plan library (often human generated). Where a plan library represents what to watch for if a plan is being attempted, a plan domain is designed for creating plans to accomplish goals. As a result, MAPRAP does not depend on human expertise to identify domain-specific recognition strategies. Likewise, this approach does not require a training set of labeled traces or a priori base rates.

Figure *1* shows our high level architecture for staging and evaluating MAPRAP (and other recognizers). We simulate a given scenario to produce a full action trace and ground truth interpretation of goals and team composition. Under the keyhole observer model (Cohen, Perrault, and Allen, 1981) used here, the recognizer has no interaction with the observed agents. The results in this paper reflect an ideal observer model with a serialized trace. Our online recognizer (MAPRAP) then infers team goals and compositions after every observation (not required). Finally, we evaluate the performance of recognition using precision, recall, and accuracy by comparing the recognizer's interpretation with the simulator's ground truth interpretation.

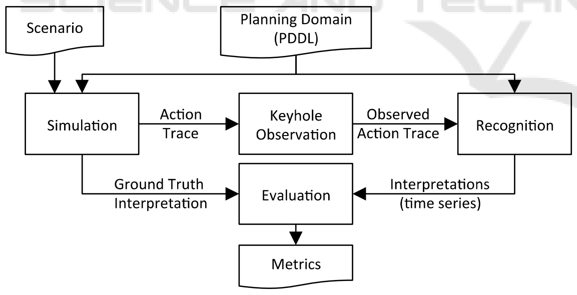

Figure 1: Our architecture uses a general planning domain to simulate and recognize multi-agent actions, enabling reliable performance evaluation.

We position this work with related research in plan recognition in Section 2. We describe our recognizer in Section 3, and our evaluation approach in Section 4. Section 5 provides baseline results for efficiency and recognition performance. This is followed by future work and conclusions.

2 RELATED RESEARCH

Multi-agent Plan Recognition (MAPR) solutions attempt to make sense of a temporal stream of observables generated by a set of agents. The recognizer's goal is to infer both the organization of agents that are collaborating on a plan, and the plan each team is pursuing. (While not addressed here, some have also included identifying dynamic teams that change over time (e.g., Banerjee, Kraemer, and Lyle 2010; Sukthankar and Sycara, 2006, 2013).) To accomplish this goal, solutions must address two challenges noted by Intille and Bobick (2001). First, the combination of agents significantly inflates state and feature spaces making exhaustive comparisons infeasible. Second, detecting coordination patterns in temporal relationships of actions is critical for complex multi-agent activities.

One approach is to use domain knowledge to identify activities indicative of team relationships. For example, Sadilek and Kautz (2010) recognized tagging events in a capture-the-flag game by detecting co-location followed by an expected effect (tagged player must remain stationary until tagged again). Sukthankar and Sycara (2006) detected physical formations in a tactical game domain and inferred cooperation to prune the search space. While practical and effective for the given domains, discovering exploitable characteristics has been a human process and similar patterns may not exist in other domains.

Generalized MAPR solutions use domain-independent recognition algorithms along with a description of the domain. Most commonly, a plan library is created that provides patterns for which a recognizer searches. For example, Banerjee, Kraemer, and Lyle (2010) matched patterns in synchronized observables, for all combination of agents, to a flattened plan library. Sukthankar and Sycara (2008) detected coordinated actions and used them to prune the multi-agent plan library using a hash table that mapped key observerable sequences for distinguishing sub-plans (i.e., last action of parent and first of sub-plan). However, it may be difficult to build a full plan library for complex domains, so others use a planning domain to guide the recognizer. Zhuo, Yang, and Kambhampati (2012) used MAX-SAT to solve hard (observed or causal) and soft (likelihood of various activities) constraints derived from the domain (action-model). In an effort to replicate the spirit of general game playing and IPC planning competitions where the algorithm is only given a general description of the problem at run-time, we use no a priori domain-specific knowledge or manually tuned libraries.

Plan Recognition as planning (PRAP) was introduced by Ramirez and Geffner in (2009) as a generative approach to single agent plan recognition that uses off the shelf planners and does not require a plan library. They convert observations to interim subgoals that the observed agent has accomplished. They synthesize plans for each goal with and without the observed subgoals, if the costs are equal then observations could be interpreted as pursuing that goal. In (Ramirez and Geffner 2010), they extended PRAP to probabilistic recognition. In the case of uniform priors, the most likely goals are those that minimize the cost difference for achieving the goal with and without explicitly meeting the observed subgoals. This research builds on PRAP for creating a generalized MAPR given a planning domain.

3 MULTI-AGENT PLAN RECOGNITION AS PLANNING

We had three key objectives for MAPRAP:

1) Perform recognition from a standardized planning domain (in PDDL) and action trace, vice a plan library. This removes the quality of the plan library as a factor in recognition performance.
2) Prune the search space intelligently for any generalized domain (without domain-specific tricks) and scale efficiently (i.e., closer to a single agent solution).
3) Accuracy of MAPR as characterized by precision, recall, and accuracy measures over time and across a wide range of randomly generated recognition instances. This provides a baseline for performance comparison.

3.1 Recognizing Multi-agent Plans

We consider a set of n agents, $A = \{A_0, A_1, \dots, A_{n-1}\}$ partitioned into m teams from a set $T = \{T_0, T_1, \dots, T_{m-1}\}$ possible teams such that each $|T_x| \geq 1$. Agents perform actions in pursuit of a team goal $G_x \in G$ for all T, where G is the set of all possible goals. In this paper, each team has one goal. Agents perform actions over time, $(1, \dots, t)$ to achieve goals. These actions, M, and environment state, E, are defined in PDDL. We define the planning domain as a tuple $D = \{A, G, M, E\}$. We define a scenario as a set of team assignments $\{(A_x, T_x) \dots (A_z, T_z)\}$ and team goals $\{(G_x, T_x) \dots (G_z, T_z)\}$ with each agent assigned to a single team and each team having a unique goal.

Our simulation component accepts both the domain and scenario, plans actions for teams, and generates a trace file. A trace consists of time-stamped observations $O = \{O_1, \dots, O_t\}$ where each includes a grounded action from D parameterized by the acting agent $a \in A$. We refer to observable actions performed by a specific agent a at time t as O_t^a. Actions that can take place concurrently (same t) are randomly ordered in the serial trace.

Our keyhole observer component interleaves the actions of all agents while maintaining action dependencies within the teams. This is also where adding noise and observation filtering can be performed, if desired. In this paper, all actions in the domain are observable, such that $O_{1\dots t}^a$ includes all actions performed by the agent from time 1 to time t, but this is not a requirement. Our system does not add "noop" actions when no action is observed for an agent at any time unless this is an action explicit in the PDDL.

The recognition component takes the domain as input and accepts a stream of observables. For each observable (alternatively each simulation time step) the recognition component outputs a set of interpretations. An interpretation is the tuple $I = (t, A_x, T_y, G_z, p, P_{T_v})$ where p is the probability (in this discrete case 0 or 1) that at time t the scenario $\{(A_x, T_v), (T_v, G_z)\}$ is true and P_{T_v} is the team plan trace on which the interpretation is based. P_{T_v} includes observerables $O_{1\dots t}^{A_x}$, and predicts future actions necessary to achieve the goal G_z

Finally, the evaluation component compares the ground truth scenario against each of the recognizer's interpretations. It scores accuracy, precision, and recall of each agent/goal combination for interpretation at each time step. We do not penalize interpretations for agents that with no observed actions up to that time step.

3.2 Extending PRAP to Teams

The single agent PRAP method of plan recognition compares the utility cost C (e.g., number of actions) of two plans for each possible goal $G_x \in G$. The first reflects only the goal G_x, the second incorporates a sequence of observations $O_{1\dots t}$ (from initial to the current time step t) expressed as subgoals that are achieved by performing the observed actions, $G_x \cap O_{1\dots t}$. When $C(G_x) < C(G_x \cap O_{1\dots t})$, the goal G_x is not supported by the observations because the observed actions increased the cost of achieving the goal. See Ramirez and Geffner's (2009) single agent PRAP for complete explanation and implementation.

We summarize performance simply as the number of plans synthesized, because CPU and clock time for planning varies greatly with domain and planner used. The worst-case number of plans synthesized for a single agent scenario (|A|=1) with interpretations at every time step is $|G| \cdot (t+1)$. If we directly apply this to any partitioning of agents into teams, the worst case is $|G| \cdot (t+1) \cdot b(|A|) \cdot |A|^2$ where $b(|A|)$ is the Bell number (web 2015) of number of agents (representing an exponential number of possible team combinations). By accepting the domain assumption that team activities are independent of other team compositions, we can reduce the worst case to $|G| \cdot (t+1) \cdot (2^{|A|}-1)$. In either case, considerable pruning is required to contain this explosion in the number of plans synthesized and make this approach tractable.

MAPRAP manages the potential agent to team assignments σ for each goal G_x^σ. We start by hypothesizing all teams compositions are pursuing all goals, until observations indicate otherwise. We then synthesize multi-agent plans using team compositions to get a base utility cost $C(G_x^\sigma)$ without observations. At each time step, $C(G_x^\sigma) < C(G_x^\sigma \cap O_{1\ldots t}^\sigma)$ identifies that the last observed action is inconsistent with the plan and MAPRAP then prunes the search space. Two variants of this algorithm (A and B) are outlined below, where an off-the-shelf planner is called to compute $C(G_x^a \cap O_{1\ldots t}^a)$ by the function "plan (goal, previously observed actions, hypothesized team composition)". The plan function internally prunes cases where no agents are hypothesized for a particular team/goal combination. The "obs (step, goal, team)" function returns the appropriate previously observed actions of the hypothesized team as subgoals. The function "report(step)" outputs the inferred agent/goal interpretations for external scoring.

In the discrete MAPRAP case, we can prune planning instances that cannot effect the interpretation. These include: any hypothesized team in time steps with no new observables for agents on the team (i.e., no explaining away alternatives), and all team/goal combinations that have been previously eliminated. MAPRAP^A implements this as outlined in Code 1.

MAPRAP^B further reduces the worst-case bounds by starting with a single team (composed of all agents) for each goal, and removing agents from compositions when their individual actions cause the utility cost to increase. While this will not work for all domains (there is a requirement for independence between the actions of agents on separate teams), the worst-case performance for MAPRAP^B (in terms of

```
//Step 1: Initialize all possible
// interpretations are feasible
#comps=2^#agents // all agent combos
hyps[#comps][#goals]=true

//Step 2: Baseline cost of plans
// for goals given no observables
for each goal in all goals
  for each team composition
    baseCost[team] goal]=
        plan(goal, null, team)

//Step 3: Process observations
// comparing costs to baseline
step=0
for each new observable action
  step++
  agent=agentFromAction(observable)
  for each goal in all goals
    for each team composition
      if(hyps[team][goal]==true)
        cost=plan(goal, obs(step,goal,
          team), team)

//Step 4: Prune compositions when
// observed actions counter plan
        if(cost > baseCost[team][goal])
          hyps[team][goal]=false

//Step 5: report metrics for time
step
  report(step)
```

Code 1: MAPRAP^A prunes team composition/goal combinations when plan given observables has a higher cost than without observables.

plans synthesized) is bound by $|G| \cdot (t+1) + |G| \cdot (|A|\text{-}1)$. The second term counts updating the baseline plan after eliminating an agent from a goal/team combination. In Code 2 the baseline cost never goes down, if reducing the number of agents in a domain could reduce cost, this strategy would fail. However, MAPRAP^B effectively reduces the number of planning jobs run closer to single agent PRAP speed.

3.3 Assumptions and Limitations

There are several aspects of MAPRAP that are not addressed in this paper, for example, alternative domains and planners, probabilistic recognition, and imperfect observer models. These will be addressed in future papers.

The given initial and discrete implementation of MAPRAP relies on two assumptions about the domain that we will resolve in future research. The first assumption is that every agent is performing towards a goal optimally and is never impeded by agents on other teams. Since team plans are synthesized independently, this also requires that the actions of different teams be independent of each other. This assumption excludes competitive domains, which is important for many applications.

```
//Step 1: Initialize all possible
// interpretations are feasible
#teams=#agents // worst case
comps[#teams][#agents]=true
hyps[#teams][#goals]=true

//Step 2: Baseline cost of plans
// for goals given no observables
for each goal in all goals
  for each team composition
    baseCost[team][goal]=
        plan(goal, null, team)

//Step 3: Process observations
// comparing costs to baseline
step=0
for each new observable action
  step++
  agent=agentFromAction(observable)
  for each goal in all goals
    for each team composition
      if (hyps[team][goal]==true)
        cost=plan(goal,obs(step,goal,
          team),team)

//Step 4: Prune agents from teams when
// their actions reduce performance
        if (cost > baseCost[team][goal])
          comps[team][agent]=false
          baseCost[team][goal]=
            plan(goal, null, team)

//Step 5: report metrics for time step
  report(step)
```

Code 2: MAPRAP^B prunes agents from teams when their actions increase the utility cost.

A second assumption, is that more agents on a team achieve a goal at least as efficiently as fewer agents, even if this means some agents simply do nothing. This may not be true for domains with communication or sequential action requirements that burden larger teams. MAPRAP relies on this condition when comparing its initial hypothesis (all agents are on the same team for all goals) to alternatives.

Other PRAP assumptions, such as finite and enumerable goals, and purposeful actions are also true of MAPRAP.

4 MAPRAP EVALUATION

MAPRAP is designed to be independent of any particular domain or planner. For this evaluation, we selected the Team Blocks domain because it is simple multi-agent adaption is well established within the MAPR community. Similarly, we chose an open source implementation of GraphPlan because it is well known and we wished to emphasize the use of an off-the-shelf planner.

4.1 A Team Blocks Domain

Team Blocks is a multi-agent adaptation of the Blocks World domain. In this domain there are a series of lettered blocks randomly stacked on a table. Each agent operates a robot gripper that can pick up one block at a time as shown in *Figure 2*. Teams are composed of 1 to |A| agents that are planning together and act collaboratively towards the same goal. Actions are atomic and include: *pickup*, *unstack* (pickup from atop another block), *put down* (on table), *stack* (put down atop another block); each action is parameterized by the block(s) acted on and agent performing the action.

We added several domain predicates to prevent agents from picking up blocks held by other agents. Since we plan teams independently, we also partitioned the blocks and goals to avoid conflicting plans. However, no information about teams (count or sizes), partitioning of blocks, or goals assignments are accessible to the recognizer.

The goal of Team Blocks is for each team to rearrange blocks into a stack in a specified sequence. Goals are random letter sequences of various lengths interpreted from bottom (on table) to up (clear). Letters are not repeated in a goal. For example, (and (ontable A) (on B A) (on C B) (clear C)) specifies the goal "ABC" pursued by Team0 in Figure *2*.

4.2 Multi-agent Plan Synthesis

We used PDDL to specify our plan domain, which enables the use of a wide range of off-the-shelf planners (e.g., those used in the IPC Planning Competitions). For this research we used an instance of GraphPlan (Pellier, 2014) with post-planning multi-agent scheduling logic. Because PDDL was

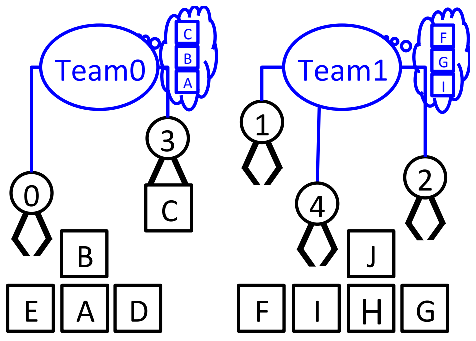

Figure 2: In Team Blocks, agents (grippers) are controlled by a team to achieve a stacking. Elements in blue are inferred by the recognizer.

not designed for multi-agent planning, every action in our domain includes a parameter of type "agent" which allows the planner to treat all available agents as a resource (i.e., it plans for the entire team). We synthesize plans for each team independently, including only agents on, or hypothesized to be on, that team. The trace outputs for all teams are interleaved by time step, and concurrent actions are shuffled to mitigate ordering effects.

We randomly generated 5180 different Team Blocks scenarios (280 were used to evaluate the less efficient MAPRAP^A) with stacks of 6-8 blocks per team (μ=6.5). We generated 1-2 teams with 1-5 agents that did not need to be balanced, but had at least one agent per team. Goals were all permutations of selected stacking orders of 4-7 blocks (μ=4.5). The number of potential goals averaged 64 for single team and 873 for two team scenarios. In the two team case, each team's partition was full sized, and possible goals included permutations across the partitions. During the generation process, we stored the ground truth team and goal assignments as a key for evaluation, but these were not available to the recognizer.

We simulated each scenario and recorded an action trace. Each trace consists of a serialized sequence of observerables identifying time step (1 to *t*), agent, and action. Traces ranged from 5 to 16 actions (μ=9.2 for single and μ=8.6 for two team scenarios). The average action parallelism (multiple agents acting in the same time step) for multi-agent scenarios was 1.4 concurrent actions. We used the action trace from initial simulation as the observables for plan recognition.

5 RESULTS

Efficiency of MAPRAP pruning is measured in terms of the count of plans synthesized, which we normalized to average number of runs per goal per time step. (*Figure 3*).

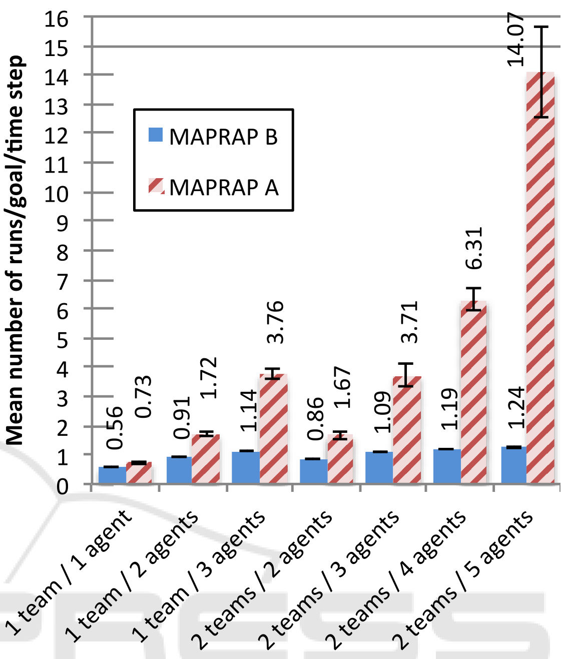

Figure 3: MAPRAP^A effectively prunes the multi-agent search space well below the exponential worst case. MAPRAP^B reduces the average runs/goal/time for scenarios to near the single agent worst-case (1.0).

Precision at each time step indicates how well recognition eliminates interpretations of the scenario that do not match ground truth. In MAPRAP, all interpretations are hypothesized correct until demonstrated to be incorrect by conflicting observations. As shown in Figure *4*, single agent scenarios require fewer observations to converge on interpretations than multi-agent scenarios.

We observed that reduced precision in the multi-agent cases reflects both fewer observations per individual agent at any time, and a large number of potential team compositions. In essence, the explanatory power of each observation is diluted across the pool of agents. As a result, it takes more observations to rule-out all feasible, but ultimately incorrect, interpretations. In fact, unlike the single agent case, most multi-agent traces ended before the recognizer converged to a single correct interpretation. We did not reduce the goal count or ensure goals diversity, which would improve precision. Since MAPRAP is an online recognizer, it is not aware does not observe the ending of a trace.

Accuracy is the ratio of correct classifications to total classifications. As shown in Figure 5, the mean accuracy of MAPRAP trails the single agent case, but demonstrates correct classifications of potential interpretations for observerables over time.

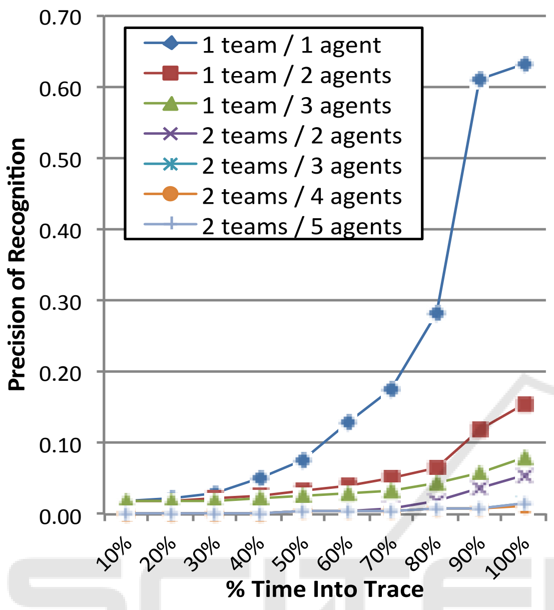

Figure 4: Precision plots show multi-agent scenarios have significantly more possible interpretations, so many more observations are required to eliminate interpretations that are consistent with observations up to that time, but incorrect.

Overall, performance of MAPRAP performance for multi-agent scenarios trailed single agent. MAPRAPA averaged 4.38 plans synthesized for each goal and time step for multi-agent, and 0.73 for single agents. MAPRAPB averaged 1.05 plans synthesized for each goal and time step for multi-agent, compared to an exponential worst case at the high end and 0.56 for single agents as a lower bound. MAPRAP recognizes multi-agent scenarios a accurately, which is driven by the ability to quickly eliminate many incorrect interpretations. However, the magnitude difference in precision between single and multi-agent scenarios reflects the large number of team composition possibilities. This indicates that few multi-agent recognition jobs converged to a single interpretation.

Recall is the measure of how well the recognizer positively classifies correct interpretations. Discrete MAPRAP, when pruning assumptions are met, has complete recall (value of 1) at every time step because it only eliminates candidate interpretations

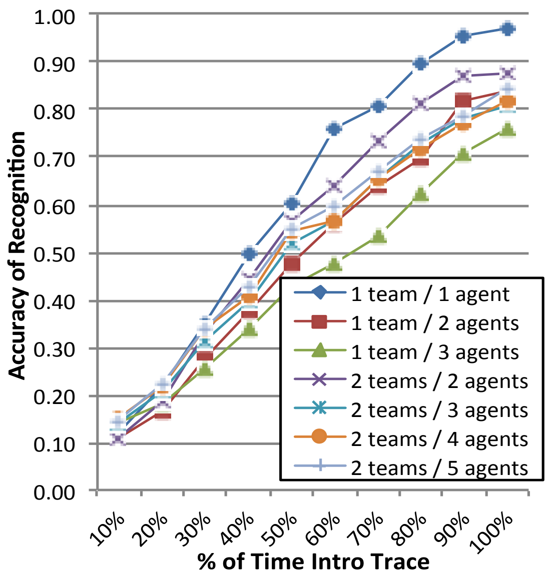

Figure 5: Accuracy metric shows single agent scenarios converge on correct interpretation faster than multi-agent scenarios.

when contradicted by observables. This results in no false negative interpretations. This was expected given that we did not implement erroneous observations, suboptimal agent action, or probabilistic recognition in this experiment.

6 FUTURE WORK

These discrete implementations of MAPRAP expose several potential areas for improvement. We are adapting additional planning domains for multi-agent benchmarking. New domains challenge the limitations of the current approach and enforce generality. One key consideration is to select/build domains that sample the relevant domain characteristic space. Also, the ability to scale from 1 agent on 1 team to n agents on n teams, ensures we the domain does not artificially limit the team composition search space, and allows us to compare performance. Similarly, Ramirez and Geffner (2009) demonstrated that satisficing and specialized planners improved speed at little cost to PRAP accuracy, making it useful for investigating larger parameter spaces. We intend to examine the use of other planners as well.

Secondly, we have implemented discrete MAPRAP. Like Ramirez and Geffner (2010), we can extend this to a probabilistic solution. Moving away from discrete decisions will introduce new efficiency challenges for which we are developing new pruning strategies. Critically, this will also better enable recognition of less optimal action traces, not currently addressed in the discrete version. We expect probabilistic interpretations will also improve precision and accuracy, but vary recall.

In addition we intend to reduce our current limitations, show the effects of observation error/loss, and reduce restrictions on inter-team interaction (e.g., competition) in future research.

7 CONCLUSIONS

In this paper we introduce a discrete version of MAPRAP, our MAPR system based on an extension to PRAP. It meets our key objective for working from a planning domain vice plan library. This enforces generalization and eliminates the dependency on human expertise in designating what actions to watch in a domain.

We show that recognizing team compositions from an online action sequence, without domain-specific tricks, greatly extends the search space. We evaluated the efficiency and performance of two MAPRAP pruning strategies on a range of Team Blocks scenarios, and established (with stated domain limitations) efficiencies nearing single agent solutions. We found we can effectively prune the search space to improve run-time independent of the planner used.

We evaluated recognition performance on a multi-agent version of the well-known Blocks World domain. We assessed precision, recall, and accuracy measures over time. This is particularly relevant as observations in multi-agent scenarios have many more possible valid interpretations than the single agent case. This in turn requires more observations to limit potential interpretations down to the single correct interpretation. Our precision and accuracy measures over time help quantify this difference.

REFERENCES

Banerjee B, Kraemer L, and Lyle J (2010) “Multi-Agent Plan Recognition: Formalization and Algorithms,” *AAAI 2010.*

Banerjee B, Lyle J, and Kraemer L (2011) “New Algorithms and Hardness Results for Multi-Agent Plan Recognition,” *AAAI* 2011.

Cohen P R, Perrault C R, and Allen J F (1981) “Beyond Question Answering,” in *Strategies for Natural Language Processing*, NJ: Hillsdale, pp. 245-274.

Genersereth M and Love N (2005) “General Game Playing: Overview of the AAAI Competition,” *AI Magazine*, vol. 26, no. 2.

Intille S S and Bobick A F (2001) “Recognizing planned, multi-person action,” *Computer Vision and Image Understanding*, vol. 81, pp. 414-445.

Kovacs D (2012) “A Multi-Agent Extension of PDDL3.1,” WS-IPC 2012:19.

McDermott D and AIPS-98 Planning Competition Committee (1998) “PDDL–the planning domain definition language”

Muise C, Lipovetzky N, Ramirez M (2014) “MAP-LAPKT: Omnipotent Multi-Agent Planning via Compilation to Classical Planning,” *Competition of Distributed and Multi-Agent Planners (CoDMAP-15).*

Pellier D (2014) “PDDL4J and GraphPlan open source implementation,” http://sourceforge.net/projects/pdd4j.

Ramirez M and Geffner H, (2009) “Plan recognition as planning,” in *Proceedings of the 21st international joint conference on Artificial intelligence.*

Ramirez M and Geffner H (2010) “Probabilistic Plan Recognition using off-the-shelf Classical Planners,” *Proc. AAAI-10.*

Sadilek A and Kautz H (2010) “Recognizing Multi-Agent Activities from GPS Data,” in *Twenty-Fourth AAAI Conference on Artificial Intelligence.*

Sukthankar G, Goldman R P, Geib C, Pynadath D V, Bui H H (2014) “Plan, Activity, and Intent Recognition Theory and Practice.” Morgan Kaufmann.

Sukthankar G and Sycara K (2006) “Simultaneous Team Assignment and Behavior Recognition from Spatio-temporal Agent Traces,” *Proceedings of the Twenty-First National Conference on Artificial Intelligence (AAAI-06).*

Sukthankar G and Sycara K (2008) “Efficient Plan Recognition for Dynamic Multi-agent Teams,” *Proceedings of 7th International Conference on Autonomous Agents and Multi-agent Systems (AAMAS 2008).*

web (2014) "Bell Numbers" *Wikipedia, The Free Encyclopedia*. Wikimedia Foundation, Inc.

Zhuo H H, Yang Q, and Kambhampati S (2012) "Action-model based multi-agent plan recognition." *Advances in Neural Information Processing Systems* 25.