Dynamic Goal Recognition Using Windowed Action Sequences

David Henri Ménager
Department of Electrical Engineering and Computer Science
University of Kansas, Lawrence, KS 66045
dhmenager@ku.edu

Dongkyu Choi
Department of Aerospace Engineering
University of Kansas, Lawrence, KS 66045
dongkyuc@ku.edu

Michael W. Floyd and **Christine Task**
Knexus Research Corporation
National Harbor, MD 20745
{michael.floyd, christine.task}@knexusresearch.com

David W. Aha
Naval Research Laboratory
Washington, DC 20375
david.aha@nrl.navy.mil

Abstract

For robots to work with humans as a team, they need to be aware of what their teammates are doing. Since it is unrealistic to expect humans to constantly communicate their goals and intentions, it is crucial for the robots to accurately and autonomously recognize their teammates' goals. Furthermore, as these human–robot teams may perform a variety of missions in dynamically changing contexts, the teammates' goals may change suddenly without warning. Historically, goal recognition systems have not directly addressed this issue, but have predominantly focused on situations with static goals. This paper presents a windowing strategy that enables goal recognition systems to detect goal changes in a fast and accurate manner. We describe this novel approach, and show its benefits through an empirical study spanning three different domains.

Introduction

Recent advances in robotics and artificial intelligence have brought a variety of assistive robots designed to help humans accomplish their goals. However, many have limited autonomy and lack the ability to seamlessly integrate with human teams. One capability that can facilitate such human-robot teaming is the robot's ability to recognize its teammates' goals, and react appropriately. This function permits the robot to actively assist the team and avoid performing redundant or counterproductive actions.

Goal recognition, like plan or intention recognition, is a difficult process, and that difficulty is compounded when goals change suddenly without warning. The majority of existing work has focused on recognizing static goals (Kautz and Allen 1986; Baker, Saxe, and Tenenbaum 2009; Sukthankar et al. 2014; Borck et al. 2015), rather than expanding goal recognition research into dynamic environments.

We describe a novel windowing approach that uses existing plan or goal recognition algorithms to rapidly detect goal changes. Our approach constrains which environmental states and actions are presented as inputs to a recognition algorithm, and uses its output to determine when a goal change occurs. This is beneficial because it allows the recognition algorithm to reason over observations related to the current goal, rather than those pertaining to previous ones.

In the following sections, we first provide background on goal recognition systems. We then explain our windowing approach and describe an empirical evaluation in three domains. Finally, we discuss related work, future research directions, and a summary of our contribution.

Background

Our windowed goal recognition system operates independently from the underlying goal recognition algorithm (i.e., it is algorithm agnostic). However, in this section we provide background on goal recognition in order to better describe how goal recognition algorithms operate. In goal recognition, the basic problem domain consists of the following:

- a set E of environment fluents;
- a state S that is a value assignment to those fluents;
- a set A of actions that describe potential transitions between states (with preconditions and effects defined over E, and parameterized over a set of environment objects O); and
- a set of disjoint goal conditions $G \subseteq S$.

If a system begins in an initial state $I \in S$ and an actor performs a sequence of actions $[a_0, a_1, a_2, \ldots, a_k]$ such that the state $a_k(...a_2(a_1(a_0(I))))$ satisfies goal $g \in G$, then we can consider $[a_0, a_1, a_2, \ldots, a_k]$ to be a *plan* to achieve goal g. The problem of goal recognition is then defined as follows:

Definition: Given a problem domain E, S, A, O, G defined as above and an action sequence q, the *goal recognition problem* is to determine, assuming q is a consecutive sub-sequence of a plan to achieve some goal g, which goal $g \in G$ is the most likely objective.

A variety of approaches may be taken to solve the goal recognition problem. Traditionally, goal recognition assumes that there is only one static goal throughout the duration of execution. Despite this assumption, researchers still implicitly make claims about the number of environmental fluents required in the recognition step, before meaningful inference is made (Sukthankar et al. 2014). In this paper we address this question directly and provide insight into where further research can be made.

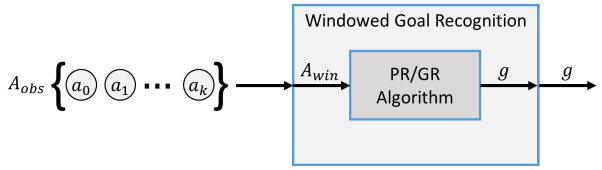


Figure 1: A diagram of windowed goal recognition wrapping around a plan recognition or goal recognition algorithm.

Windowed Goal Recognition

Since traditional goal recognition algorithms are designed to recognize static goals, when goal changes occur they provide inconsistent observations to the algorithms (i.e., some observations of the old goal and some of the new goal). In most situations, this results in the algorithms selecting the goal with the strongest evidence to support it. If there is significant evidence of the original goal (e.g., the agent had been pursuing that goal for a long duration), the algorithms may be unable to detect the new goal until there it has an equivalent amount of evidence. For example, consider an observed action sequence $[a_0, a_1, \ldots, a_m]$. If $[a_0, \ldots, a_{m-1}]$ were performed when pursuing goal g_1 whereas $[a_m]$ was performed when pursuing goal g_2, it is difficult to recognize g_2 as the goal because the majority of actions are related to g_1. If m is larger (i.e., the agent pursued g_1 for a long time), many more observations will be necessary before g_2 can be successfully identified. To remedy this, we present a windowing strategy that uses a subset of the action sequence and restricts the input to the goal recognition algorithms.

Our approach operates by wrapping around existing goal recognition algorithms. Figure 1 shows a windowing module wrapping around a goal recognition component. This is beneficial because it does not require any modifications to the algorithms, and goal recognition algorithms can be used interchangeably. The windowed goal recognition system observes an agent performing a sequence of actions $A_{obs} = [a_0, a_1, \ldots, a_k]$ and creates a sub-sequence A_{win} of A_{obs} that only contains the most recent observed actions. For example, if a window size of 3 is used, $A_{win} = [a_{k-2}, a_{k-1}, a_k]$. A_{win} is then used as input to the goal recognition algorithm, rather than the entire sequence of actions. Figure 2 shows windows that grow to a maximum size of five (i.e., w1-w5) and shift (i.e., w6-w7) using an example action sequence.

Using windowed goal recognition is advantageous for three primary reasons. First, it does not depend on any particular plan recognition algorithm. Any algorithm that takes as input a sequence of actions and outputs a recognized goal can be used. This approach can also be used for algorithms that reason over sequences of both actions and environment states, with the only modification being that windows operate over action-state sequences rather than action sequences. Second, the approach helps remove older observations that may be related to previous goals. This allows for quicker goal change detection than if the entire action sequence is used. Third, the windowing approach provides an upper bound on the number of actions provided as input to the goal recognition algorithms. For real-time applications, this helps to constrain the time spent on goal recognition.

Empirical Evaluation

To assess the benefits of our windowed goal recognition strategy, we performed experiments with tasks in three domains. In each, we consider three hypotheses:

H_1**:** Windowing improves the performance of goal recognition systems in dynamic environments

H_2**:** Windowing mechanism makes goal recognition performance independent of the state–action sequence length

H_3**:** Windowing works across different domains

In the first hypothesis we use two instances of the same goal recognition system. Our windowing procedure wraps around the first instance, and the second is left as-is. We compare the windowed and non-windowed performance of the goal recognition system and observe whether the windowing procedure significantly increases its goal change detection accuracy when compared to the non-windowed version. In each domain, we analyzed the performance of the windowed and non-windowed versions, while varying the timing of a goal change within the state–action sequence.

For the second hypothesis we examine the performance of a windowed goal recognizer subject to varying goal change indexes, and varying number of goal changes in the executed plan. We are interested in quantifying how long the system takes to recover from incorrect goal detections, and measuring the flexibility of the system under rapid goal changes. For completeness we also examine the performance without the window and compare the performance.

Lastly, we are interested in the generalization ability of the windowing approach. We mean to create a system that allows goal recognition systems to perform comparably in different domains under the same system parameters. In this section, we present the goal recognition algorithm and briefly describe the three domains. Following this, we detail the experimental setup, and present and discuss the results.

SET-PR

Single-agent Error-Tolerant Plan Recognizer, SET-PR (Vattam, Aha, and Floyd 2014), is a fault-tolerant graph-matching technique for goal recognition. We use this system as our test case for which we built our windowing approach because it gracefully handles missing and erroneous actions. SET-PR uses a case-based approach that leverages a library of canonical plans for achieving each of the goals in the domain. Both library plans and input action sequences are represented using Action Sequence Graphs (ASGs) (shown in Figure 3). These graphs enable SET-PR to encapsulate information about the patterns of relationships between actions and objects in the environment independently of the exact action ordering. The ASG represents objects in the environment and executed actions as nodes. Labeled directed edges are added to connect action nodes and the nodes representing their parameter objects.

If an action sequence demonstrates similar patterns of action/object connections to a canonical plan in the library, then it may be a sub-sequence of that plan. The input action sequence is matched against canonical plans in the library using a relaxed (multi-graph handling) version of the

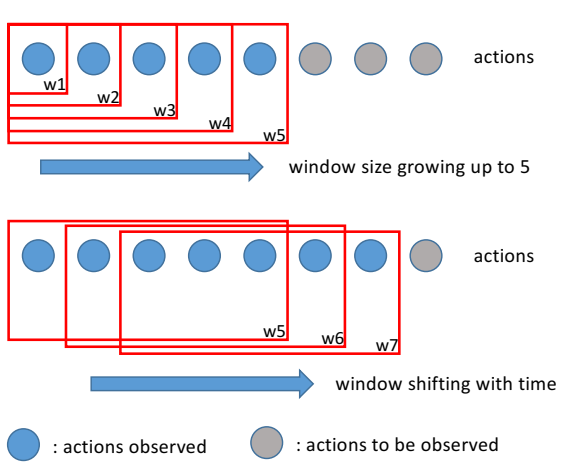


Figure 2: A sample window growing and shifting on an arbitrary action sequence.

VF2 graph-matching algorithm (Cordella et al. 2004). SET-PR returns the similarity scores for each of the candidate library plans, with the most similar plan being selected as the most likely plan of the observed agent. The goal associated with the selected plan is extracted and used by the windowing approach to detect goal changes. Further details on SET-PR may be found in Vattam, Aha, and Floyd (2014).

Evaluation Domains

We used three different domains for empirical evaluation. The first one is a custom military domain called *Autonomous Squad Member* (ASM) (Gillespie et al. 2015). This domain has a simulated robotic agent that follows its human squad members on combat reconnaissance missions. In the scenarios we used, there are three human soldiers and a robot in each squad. The soldiers can perform a variety of actions like moving to a waypoint, following another soldier, gesturing or speaking aloud to a soldier, assuming a defensive position, and taking a posture, in rich settings that include different entities, weapons, and geographic features. The robot can observe the state and the actions of its teammates and choose among its own possible actions relevant to the situation it has determined based on that information.

The other two domains are benchmark domains from the International Conference on Automated Planning and Scheduling (ICAPS) International Planning Competition (IPC). The first one of these is *Rovers*, a domain with simplified planetary rover missions. Each scenario happens on a map represented as a connected graph, on which the rover can move, sample soil, sample rock, drop collected sample at its base, and communicate the test result. The second domain, *Childsnack*, involves an agent that serves sandwiches to children. Depending on a child's dietary restriction (i.e., gluten or gluten-free), the agent makes a sandwich with appropriate contents, puts it on a tray, moves the tray to a table, and serves the sandwich to the child.

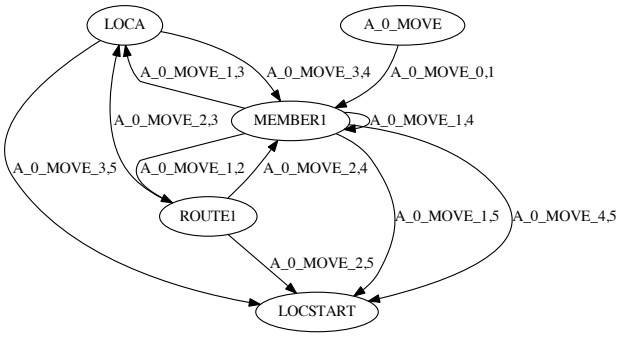


Figure 3: A sample Action Sequence Graph (ASG) from Autonomous Squad Member domain.

Experimental Setup

In order to create plan recognition problems for evaluation, we generated state–action sequences for various goals in each domain. For example, in the ASM domain, we used three scenarios that involve a single goal (investigating a route), and three other scenarios that involve two goals (investigating a route and responding to a sniper). For both the one-goal and two-goal variations, two scenarios were used to generate canonical plans for SET-PR's plan library, and the other was used to generate testing plans.

Although we did not have the ability to generate more problem scenarios in the ASM domain, we were able to randomly generate valid problem descriptions for the two IPC domains. Once we have problems generated, we used the Fast Downward Planner (Helmert 2006) to compute plans for these problems and get state–action sequences. In both domains, the agent was placed in one of ten randomly generated environments. To generate the canonical plan library for SET-PR, we created random initial states and ran the planner for different goals to generate the corresponding state–action sequences. In Rovers, the goals were to collect *soil* or *rock*, whereas in Childsnack the goals were to make a *gluten* or *gluten-free* sandwich.

After we generated the plan library in this manner, a similar strategy was used to generate test cases, except that the test cases contained goal changes that our system needed to detect. For each test case, the agent starts with an initial goal and then changes its goal during execution (with the timing of the goal change varying among test cases). Overall, we used this process to generate 100 different test cases per domain. In our experiments, we labeled our results based on the order of the goal change. Namely, a'0' denotes a goal change from rock to soil (Rovers) or gluten to gluten-free (Childsnack). A '1' denotes a goal change from soil to rock (Rovers) or gluten-free to gluten (Childsnack).

During each experiment we measured:

1. The percentage of runs the algorithm detects the correct initial goal
2. The percentage of runs the algorithm detects the correct final goal
3. The mean distance between the detected goal change and actual goal change

4. The mean number of actions needed to converge to true initial goal
5. The mean number of actions needed to converge to true final goal

In total, we performed experiments in four different conditions for each of the IPC domains. Two of them were windowed, and the others were not. For both windowed and non-windowed cases, we performed experiments with goal changes in each direction (i.e., those labeled '1' and those labeled '0'). The window size was kept constant at five for all domains.

Experimental Results

In this section, we report results that show strong evidence for our three hypotheses. Figure 4 provides results from two sample runs in the ASM domain. In both cases, the goal changes happen relatively close to each other at third, seventh, and ninth actions for the first one, and at third, fourth, and sixth actions for the second one. These cases describe two team members reacting to an unanticipated enemy sniper while they are investigating a route. Initially, the members think they have eliminated the threat after having engaged the enemy, and resume investigating the route. But they quickly realize that they were wrong after the sniper starts firing on them again. As shown in the graphs, the windowed SET-PR reacts to the goal changes fairly well when compared to the non-windowed SET-PR (not shown) that effectively saturates and does not provide useful results. But sometimes the former does not react fast enough because there still is too much evidence for the old plan in the window. We believe that the optimal window size is scenario-dependent and it is related to the minimum distance between goal changes.

Tables 1 and 2 summarize the Rovers and Childsnack results. Overall, the windowed SET-PR shows superior performance in detecting the goal change compared to the non-windowed counterpart, as evidenced by the percentages of correct detection for the final goals (92% vs. 12% for Rovers and 100% vs. 11% for Childsnack). This, along with the results from the ASM domain, proves the first hypothesis, H_1, that windowing approach improves the goal recognition performance in dynamic environments with one or more goal changes. Interestingly, in the Rovers domain, both windowed and non-windowed systems do a poor job detecting

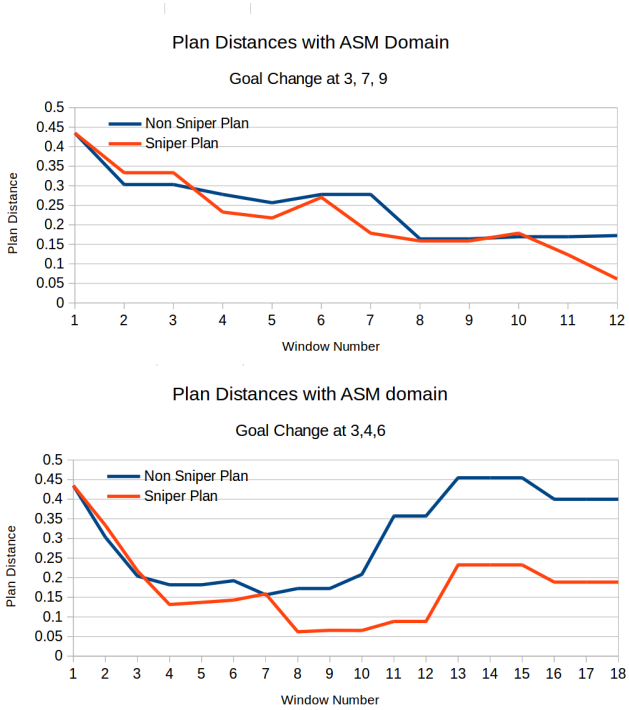


Figure 4: Windowed goal recognition performance for the ASM domain. For each point on the horizontal axis, the plan with the lowest plan distance is what SET-PR estimates to be the current plan.

the initial goals at the beginning of the plan execution. This is largely a result of how closely related the two plans are in this domain (soil and rock). The only distinguishing action between these plans is whether a soil or rock sample is taken. All the movement and collection actions are shared across both plans. So, in the beginning, it is difficult to distinguish the true plan of the agent. Also, this delay in detecting the true goal is responsible for high average distances between the detected and actual goal changes. SET-PR often detects a goal change at this stage, but this is one to correct its initial error, not a true goal change. This highlights the fact that the performance of the windowed system is still highly reliant on that of the underlying goal recognition algorithm. But, since the approach is algorithm-agnostic, a new algorithm can be used as necessary.

Table 1: Performance over all runs in Rovers domain.

Windowed Rovers1	
Percent Correct Initial Goal Detected	13.86
Percent Correct Final Goal Detected	92.08
Mean Dist. Btn. Detected & Actual Goal Change	21.85
Mean Actions to Converge To True Initial Goal	7.07
Mean Actions to Converge To True Final Goal	1.04
Not Windowed Rovers1	
Percent Correct Initial Goal Detected	17.82
Percent Correct Final Goal Detected	11.88
Mean Dist. Btn. Detected & Actual Goal Change	21.73
Mean Actions to Converge To True Initial Goal	7.24
Mean Actions to Converge To True Final Goal	3.34

Table 2: Performance over all runs in Childsnack domain.

Windowed Childsnack0	
Percent Correct Initial Goal Detected	100.0
Percent Correct Final Goal Detected	100.0
Mean Dist. Btn. Detected & Actual Goal Change	0.89
Mean Actions to Converge To True Initial Goal	1.0
Mean Actions to Converge To True Final Goal	1.0
Not Windowed Childsnack0	
Percent Correct Initial Goal Detected	100.0
Percent Correct Final Goal Detected	10.89
Mean Dist. Btn. Detected & Actual Goal Change	10.89
Mean Actions to Converge To True Initial Goal	9.32
Mean Actions to Converge To True Final Goal	18.81

Table 3: Windowed performance in the Rovers domain, partitioned by when the goal change occurred.

Goal Change Percent Range: [0, 30]	
Percent Correct Initial Goal Detected	35.48
Percent Correct Final Goal Detected	100.00
Mean Dist. Btn. Detected & Actual Goal Change	3.77
Mean Actions to Converge To True Initial Goal	2.61
Mean Actions to Converge To True Final Goal	1.0
Goal Change Percent Range: [30, 60]	
Percent Correct Initial Goal Detected	6.67
Percent Correct Final Goal Detected	100.00
Mean Dist. Btn. Detected & Actual Goal Change	19.57
Mean Actions to Converge To True Initial Goal	7.53
Mean Actions to Converge To True Final Goal	1.0
Goal Change Percent Range: [60, 100]	
Percent Correct Initial Goal Detected	2.50
Percent Correct Final Goal Detected	80.0
Mean Dist. Btn. Detected & Actual Goal Change	37.58
Mean Actions to Converge To True Initial Goal	10.18
Mean Actions to Converge To True Final Goal	1.10

Table 4: Windowed performance in the Childsnack domain, partitioned by when the goal change occurred.

Goal Change Percent Range: [0, 30]	
Percent Correct Initial Goal Detected	100.0
Percent Correct Final Goal Detected	100.0
Mean Dist. Btn. Detected & Actual Goal Change	0.65
Mean Actions to Converge To True Initial Goal	1.0
Mean Actions to Converge To True Final Goal	1.0
Goal Change Percent Range: [30, 60]	
Percent Correct Initial Goal Detected	100.0
Percent Correct Final Goal Detected	100.0
Mean Dist. Btn. Detected & Actual Goal Change	1
Mean Actions to Converge To True Initial Goal	1
Mean Actions to Converge To True Final Goal	1
Goal Change Percent Range: [60, 100]	
Percent Correct Initial Goal Detected	100.0
Percent Correct Final Goal Detected	100.0
Mean Dist. Btn. Detected & Actual Goal Change	1
Mean Actions to Converge To True Initial Goal	1
Mean Actions to Converge To True Final Goal	1

As shown in Tables 3 and 4, where we partitioned the windowed case results based on when the goal change occurs (i.e., in the first 30% of actions, after the first 30% but before 60% of actions, or after 60% of actions), the performance of the windowed goal recognition was not affected by the timing of the goal change. For the reasons described earlier, the system might (in the Roverst domain) or might not (in the Childsnack domain) detect the initial goals correctly in the beginning of the sequences. But nonetheless, the system recovers from its erroneous judgment and correctly detects when a goal change happens. Moreover, the windowed system is not affected by the lengths of the plans showing that it scales nicely as we hypothesized in H_2.

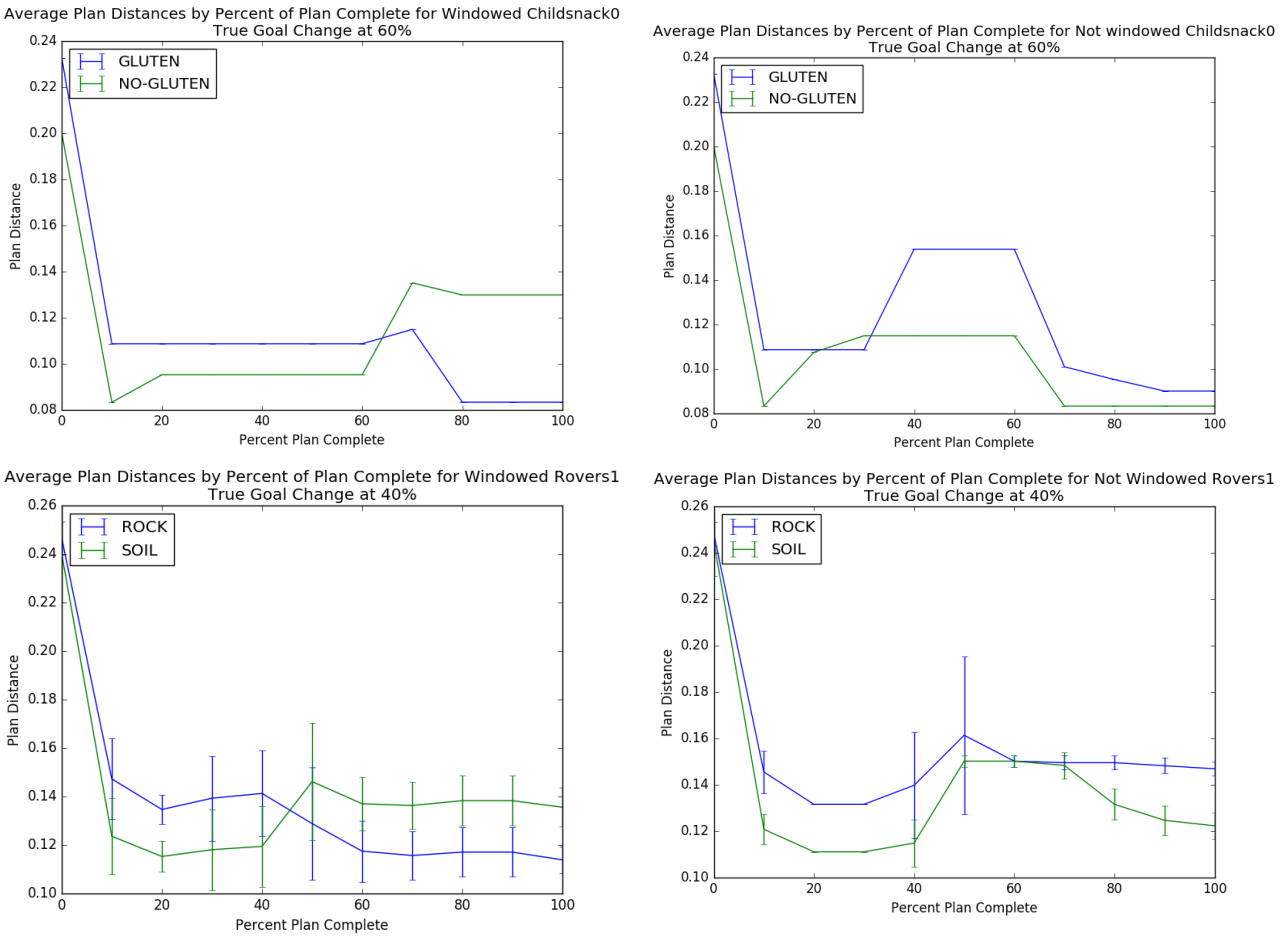
The plots shown in Figure 5 look to demonstrate what is occurring in SET-PR at several sample goal change locations (i.e., what percent of actions are observed before a goal change). The plots show the average distance to the most similar plan in the plan library of each type. For the windowed case (on the left column), there is a clear switch after the goal change such that the plan that was originally most similar is now less similar than the new goal. However, non-windowed SET-PR (on the right column) does not have such a clear differentiation and often never determines the new goal correctly. The results further validate our first hypothesis, showing that applying the windowing strategy to the SET-PR algorithm allows it to quickly detect goal changes.

In the Childsnack domain, plans are more distinguishable since it begins by making the sandwiches early in the plan (i.e., the more distinguishable actions). Furthermore, there are fewer generic movement commands taking a significant portion of the action sequence. Non-windowed SET-PR incorrectly detects goals changes much earlier than they actually occur. Similarly, it must observe significantly more actions before converging to the correct initial or final goal. These results provide some evidence that highlight the rigidity of the non-windowed approach. Although it detects the initial goal reasonably well in some instances (i.e., comparable performance to the windowed approach), it performs significantly worse than our windowed approach at detecting goal changes.

Looking more closely at the Rovers domain results in Figure 5, windowed SET-PR has a standard deviation that remains almost constant at a (mostly) higher-level than the non-windowed case. This is because the windowed system gains increased flexibility by using smaller subset of state–action sequence, which allows it to react to dynamic environments. But, it could also mean that the system makes more errors in goal recognition by being too quick to label a goal change. This is why the window size needs to be carefully set at a reasonable level, avoiding mistakes in plan recognition that can happen where the error bars overlap like in Figure 5a.

Note that the recognition of goal changes in the non-windowed cases is significantly slower and it is often unable to detect a goal change occurred. For example, take the non-windowed SET-PR performance in Figure 5b. The true goal change occurs at 20% of the way through the plan. SET-PR does not recognize this until 35 to 40% of the plan has already been executed. After passing that point, the plans begin to revert back to their original distances, resulting in SET-PR never consistently selecting the correct goal. The reason for the delayed reaction time is because of the history of past actions that is used during goal recognition. Also, recall that the observed actions are represented in an ASG. Although some ordering information is retained, as the system observes more actions, it will contain information for both types of plans. Thus, the observed ASG will have many nodes and edges that match many different plans. As a result, a goal change may not add enough information for detection.

Combining all the results we have covered so far, we found that the windowing approach works across all these domains, proving our third and final hypothesis. Hence, we demonstrated that our novel approach satisfies all three hypotheses presented. In the next section, we place our work within the larger body of scientific knowledge on goal recognition for discussions.



(a) Windowed SET-PR performance

(b) Non-windowed SET-PR performance

Figure 5: Goal recognition performance for Childsnack and Rover domains. Note that the standard use of SET-PR gives no useful information regarding the goal change, while the windowed case shows precise tracking of goal changes.

Related Work

Prior work on goal recognition (Kautz and Allen 1986; Baker, Saxe, and Tenenbaum 2009; Sukthankar et al. 2014; Borck et al. 2015) explored a variety of techniques. However, most focused on recognizing a single, static goal.

Baker et al. (2009) investigate goal recognition in the context of changing goals. Similar to our research, they proceed from the principle of rationality and assume that agents plan approximately rationally to achieve their goals. They represent this with a Bayesian network model which states that an agent's actions are determined by its goal and the environment. To infer an agent's current goal g, they estimate the likelihood of the observed actions given prior information about g. One of the models that they create accounts for goal changes; it was particularly effective at identifying the mental state of the agents. The authors remark about the need to restrict the number of actions used for detection.

Our work differs in three primary ways. First, their best performing model could detect only one goal change, while ours can detect any number of dynamic goal changes. Second, our approach is system-agnostic; it can be used with a large variety of goal recognition algorithms. Finally, we describe a novel technique that can be integrated with many existing goal recognition algorithms.

Min et al. (2016) perform goal recognition using a Long Short-Term Memory, a variant of a recurrent neural net in an open-world game domain. Because of the lack of explicit directions in games of this type, they relaxed the assumption that agents act in somewhat rational ways to achieve their goals. Often times, humans must explore the space without well-defined goals before directing their actions to more meaningful objectives. Min et al. report that their method surpasses state of the art performance in goal recognition. The network takes for input a sequence of actions. The network predicts the probability that an agent is achieving a particular goal given the prior belief about the agent's previous goal.

In order to train this network this method required a data set with 77,182 player actions from 893 achieved goal demonstrations. Each demonstration containing around 86 player actions. Many times the amount of data required to

train learning methods like this deter people from using them. Case-based reasoning approaches have their own limitations, but they only require a few cases in the case base to make predictions. New cases are easily incorporated into the system that improve the quality of goal detections. We do not claim that our method is superior to Min et al., because our experiments do not operate in the same domians, but we do provide evidence that our method is also a first-class strategy for goal recognition.

Borck et al. (2015) use a case-based approach for plan/goal recognition to control unmanned air vehicles (UAVs) in (simulated) beyond-visual-range air combat scenarios. The task for these UAVs is to estimate an adversary's current policy (i.e., a function that maps states to actions) using prior information about the adversary's goal, which is provided in a mission briefing. Their plan recognizer detects when its prior information about an adversary is incorrect, and update its assumptions accordingly. This can increase the accuracy of policy recognition.

This differs from our work in that Borck et al. assume that the agent has a single, possibly unknown (or incorrectly estimated), goal. In contrast, we assume that there will be multiple (temporally-disjoint) goals that an adversary will pursue during a mission scenario.

Finally, Keren et al. (2016) present a framework for offline recognition of static goals under partial observability. In this work, the actors are allowed to act optimally, or suboptimally. The task is to have an observer infer the actor's goal, subject to missing action information The method was tested in three classic domains, blocksworld, logistics, and grid navigation.

Future Work

We are particularly interested in improving on the SET-PR graph matching algorithm and creating abstractions on the action sequence graphs to yield event sequence graphs. This would provide a better underlying goal recognition algorithm that operates under our windowing approach. Since our method uses existing goal recognition algorithms, any improvements to the algorithms are expected to directly improve the performance of our windowing approach.

Specifically, we wish to evaluate our work on a goal recognition algorithm that has more robust capabilities for dealing with plans with generic actions and allows abstracting events from the low-level action information. In the case of SET-PR, this would potentially reducing the number of nodes and edges in the graph while still keeping qualitative and quantitative information about what is being observed.

We also plan to evaluate our windowed approach using a wider variety of goal recognition algorithms. Our approach was only evaluated using a single algorithm, but we plan to demonstrate that our approach is algorithm-agnostic by replicating our results with other goal recognition algorithms. This will demonstrate that our approach is a simple and effective way to address dynamic goal changes.

Conclusions

In this paper, we described a novel windowing strategy that can be applied to an existing goal recognition system. This approach was designed to detect goal changes faster and more reliably, and we have demonstrated improved performance over the SET-PR algorithm. We have shown that the windowing approach works in three different domains, Autonomous Squad Member, Rovers, and Childsnack. Specifically, the windowing procedure dramatically improves the goal change recognition capabilities. Lastly, we placed our work within the larger context of work in plan/goal recognition, and presented our future work.

References

Baker, C. L.; Saxe, R.; and Tenenbaum, J. B. 2009. Action understanding as inverse planning. *Cognition* 113(3):329–349.

Borck, H.; Karneeb, J.; Floyd, M. W.; Alford, R.; and Aha, D. W. 2015. Case-based policy and goal recognition. In *Proceedings of the Twenty-Third International Conference on Case-Based Reasoning*, 30–43. Springer.

Cordella, L. P.; Foggia, P.; Sansone, C.; and Vento, M. 2004. A (sub) graph isomorphism algorithm for matching large graphs. *IEEE transactions on pattern analysis and machine intelligence* 26(10):1367–1372.

Gillespie, K.; Molineaux, M.; Floyd, M. W.; Vattam, S. S.; and Aha, D. W. 2015. Goal reasoning for an autonomous squad member. In *Goal Reasonning: Papers from the ACS Workshop*. Georgia Institute of Technology, Institute for Robotics and Intelligent Machines.

Helmert, M. 2006. The fast downward planning system. *Journal of Artificial Intelligence Research* 26:191–246.

Kautz, H. A., and Allen, J. F. 1986. Generalized plan recognition. In *Proceedings of the Fifth National Conference on Artificial Intelligence*, 32–38. AAAI Press.

Keren, S.; Gal, A.; and Karpas, E. 2016. Privacy preserving plans in partially observable environments. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, 3170–3176. AAAI Press.

Min, W.; Mott, B.; Rowe, J.; Liu, B.; and Lester, J. 2016. Player goal recognition in open-world digital games with long short-term memory networks. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, 2590–2596. AAAI Press.

Sukthankar, G.; Geib, C.; Bui, H. H.; Pynadath, D.; and Goldman, R. P. 2014. *Plan, activity, and intent recognition: theory and practice*. Morgan Kaufmann.

Vattam, S. S.; Aha, D. W.; and Floyd, M. W. 2014. Case-based plan recognition using action sequence graphs. In *Proceedings of the Twenty-Second International Conference on Case-Based Reasoning*, 495–510. Springer.